

**LexGrid Vocabulary Services for caBIG™**  
***(LexBIG, Phase I)***  
**Use Case Specification**  
**Version 1.0**

**MAYO CLINIC DIVISION OF BIOMEDICAL INFORMATICS,  
MAYO CLINIC COLLEGE OF MEDICINE**

**Wednesday, August 31, 2005**

---

**Document Change Record**

Version Number	Date	Description
0.5	July 27, 2005	<ul style="list-style-type: none"><li>• Initial Draft</li></ul>
1.0	August 31, 2005	<ul style="list-style-type: none"><li>• Initial version available for public review.</li></ul>

<b>USE CASE SPECIFICATION .....</b>	<b>I</b>
<b>DOCUMENT CHANGE RECORD.....</b>	<b>II</b>
<b>1 INTRODUCTION.....</b>	<b>1</b>
1.1 OVERVIEW .....	1
1.2 BACKGROUND AND SCOPE .....	1
1.3 TERMINOLOGY .....	1
1.4 RELATED DOCUMENTS.....	2
<b>2 ACTORS.....</b>	<b>2</b>
<b>3 USE CASES.....</b>	<b>3</b>
3.1 INSTALLATION.....	3
<b>3.1.1 Install Software .....</b>	<b>3</b>
3.1.1.1 Acquire Software .....	3
3.1.1.2 Install Software .....	4
3.1.1.3 Configure Software Installation .....	4
3.1.1.4 Bind to caCORE Server.....	5
3.1.1.5 Bind to Language Runtimes.....	5
3.1.1.6 Bind to Remote Services.....	5
3.1.1.7 Validate Software Installation.....	6
3.1.1.8 Install Content.....	6
3.2 CONTENT CONVERSION AND LOAD .....	7
<b>3.2.1 Publishing and Acquisition .....</b>	<b>8</b>
3.2.1.1 Publish Vocabulary Revision.....	8
3.2.1.2 Acquire Vocabulary .....	8
3.2.1.3 Validate Vocabulary Content.....	8
<b>3.2.2 Import.....</b>	<b>8</b>
3.2.2.1 Import to LexBIG Repository .....	8
3.2.2.2 Import Revision to LexBIG Repository .....	9
3.2.2.3 Build Index .....	9
<b>3.2.3 Export.....</b>	<b>10</b>
3.2.3.1 Export to LexBIG XML.....	10
3.3 MAINTENANCE .....	11
<b>3.3.1 Software Updates .....</b>	<b>11</b>
3.3.1.1 Acquire Software Update.....	11
3.3.1.2 Install Software Update.....	12
3.3.1.3 Reconfigure Software .....	12
3.3.1.4 Submit Feedback.....	12
<b>3.3.2 Maintain Vocabulary Registry.....</b>	<b>13</b>
3.3.2.1 Add New Vocabulary .....	14
3.3.2.2 Add Vocabulary Revision.....	14
3.3.2.3 Retire Vocabulary Revision .....	14
3.3.2.4 Browse Available Vocabularies.....	15
3.3.2.5 Register for Update Notification.....	15
3.3.2.6 Revise or Remove Update Notification .....	15
<b>3.3.3 Content Change Requests .....</b>	<b>16</b>
3.3.3.1 Create Change Request .....	16
3.3.3.2 Submit Change Request.....	17
3.3.3.3 Review Change Request .....	17
3.3.3.4 Notify User of Decision .....	17
3.3.3.5 Update Vocabulary .....	17
<b>3.3.4 Create Change Request (expanded) .....</b>	<b>18</b>
3.3.4.1 Modify Existing Concept.....	18
3.3.4.2 Create a New Concept .....	18
3.3.4.3 Split a Concept.....	19
3.3.4.4 Merge two Concepts.....	19

3.3.4.5	Retire a Concept.....	19
3.3.4.6	Free text Submission.....	19
<b>3.3.5</b>	<b>Create Pick List.....</b>	<b>20</b>
3.3.5.1	Establish Context.....	20
3.3.5.2	Acquire List of Possible Codes.....	20
3.3.5.3	Organize list for ease of use.....	21
3.3.5.4	Select Appropriate Representative Terms.....	21
3.3.5.5	Assign Shortcuts and other Selection Devices.....	21
3.3.5.6	Save Picklist for Subsequent Use.....	21
<b>3.4</b>	<b>RUNTIME ACCESS.....</b>	<b>22</b>
<b>3.4.1</b>	<b>EVS Access: Annotate a Data Model using Semantic Connector.....</b>	<b>22</b>
3.4.1.1	Create data model.....	22
3.4.1.2	Execute semantic connector.....	23
3.4.1.3	Resolve EVS Service.....	23
3.4.1.4	Invoke caCORE EVS API.....	23
3.4.1.5	Review spreadsheet of class and attribute mapping.....	24
3.4.1.6	Submit spreadsheet to NCI.....	24
<b>3.4.2</b>	<b>Programmatic Access to LexBIG API.....</b>	<b>25</b>
3.4.2.1	Establish Connectivity.....	25
3.4.2.2	Remote Reference.....	26
3.4.2.3	Direct Language Reference.....	26
3.4.2.4	Establish Session.....	26
3.4.2.5	Method Invocation.....	26
3.4.2.6	Retrieve uniquely specified concept.....	27
3.4.2.7	Query concepts by criteria.....	27
3.4.2.8	DAG Traversal.....	27
3.4.2.9	Combinatorial Access.....	27
3.4.2.10	Server and Vocabulary Metadata Access.....	28
<b>3.4.3</b>	<b>Access to History and Version Information.....</b>	<b>29</b>
3.4.3.1	Request History for Uniquely Specified Concept.....	29
3.4.3.2	Request Change Report for Vocabulary Version.....	29
3.4.3.3	Determine State of Individual Concept.....	30
3.4.3.4	Retrieve Available Versions of a Code System.....	30
<b>3.5</b>	<b>SECURITY, INTEGRITY, AND ACCESS CONSTRAINTS.....</b>	<b>31</b>
<b>3.5.1</b>	<b>Passive Model.....</b>	<b>32</b>
3.5.1.1	Request Terms of Use.....	32
<b>3.6</b>	<b>IDENTIFIER RESOLUTION.....</b>	<b>33</b>
<b>3.6.1</b>	<b>Identifier Resolution.....</b>	<b>33</b>
3.6.1.1	Locate Directory Aware Service Node.....	33
3.6.1.2	Establish Application Context.....	34
3.6.1.3	Access LexBIG content.....	34
3.6.1.4	Register Service with Directory.....	34
3.6.1.5	Resolve External Identifier References.....	34
<b>3.7</b>	<b>COLLABORATION.....</b>	<b>36</b>
<b>3.7.1</b>	<b>caTIES Retrieval Application.....</b>	<b>36</b>
3.7.1.1	Identify Research Question.....	37
3.7.1.2	Identify Search Concepts for Data Retrieval.....	37
3.7.1.3	Map Search Concepts to Vocabulary Concept Codes.....	38
3.7.1.4	Browse Vocabulary for Search Concept.....	38
3.7.1.5	Search Vocabulary for Search Concept.....	38
3.7.1.6	Formulate Query Logic.....	39
<b>3.7.2</b>	<b>caTIES Execute Query.....</b>	<b>40</b>
3.7.2.1	Execute Query.....	40
3.7.2.2	Save Query.....	40
3.7.2.3	Review Query results.....	40
<b>3.7.3</b>	<b>caTIES Annotation.....</b>	<b>41</b>
3.7.3.1	Prepares Document.....	41
3.7.3.2	Annotate document with vocabulary concepts.....	42
3.7.3.3	Maps textual information to vocabulary concepts.....	42
3.7.3.4	Inference concepts based using text and DL of vocabulary.....	42
3.7.3.5	Persists annotation to permanent storage.....	42
<b>4</b>	<b>NOTES.....</b>	<b>43</b>

# 1 Introduction

---

## 1.1 Overview

This document presents use cases for the LexGrid Terminology Services (LexBIG) project, which is being developed by the Mayo Clinic Division of Biomedical Informatics for caBIG. The goal of the project is to build a vocabulary server accessed through a well-structured application programming interface (API) capable of accessing and distributing vocabularies as commodity resources. The server is to be built using standards-based and commodity technologies. Primary objectives for the project include:

- Provide a robust and scalable open source implementation of EVS-compliant terminology services. The API specification will be based on but not limited to fulfillment of the caBIO EVS API. The specification will be further refined to accommodate changes and requirements based on prioritized needs of the caBIG community.
- Provide a flexible implementation for terminology storage and persistence, allowing for alternative mechanisms without impacting client applications or end users. Initial development will focus on delivery of open source freely available solutions, though this does not preclude the ability to introduce commercial solutions (e.g. Oracle).
- Provide standard tooling for load and distribution of vocabulary content. This will involve support of standardized representations in UMLS Rich Release Format (RRF) and the OWL web ontology language. (Vocabulary editing, vocabulary submission, and vocabulary cross-linking are out of scope for this aim.)

## 1.2 Background and Scope

This document draws on use cases from the following sources:

- *Vocabulary service providers.* Describes organizations currently supporting externalized API-level interfaces to terminological content for the caBIG™ community. Practically speaking, this describes the NCI EVS and caCORE teams. Current API-level interfaces include the caCORE EVS and Metaphrase APIs. Several discussions have occurred with the EVS team to refine requirements regarding backward compatibility with existing services and data interchange formats.
- *Vocabulary integrators.* Describes organizations within the caBIG™ community that desire to integrate new terminological content or relations to be served to the caBIG™ community. This includes domain-specific stakeholders, such as the University of Pittsburgh Medical Center (SPIN and caTIES semantic cross-linking), Cancer Research Center of Hawaii (nutrition ontology), and Jackson Labs (mouse anatomy).
- *Vocabulary users.* Describes the caBIG™ community in general. Specific sources of information from this larger community include the V/CDE vocabulary surveys.

***NOTE: Use cases described in this document are not presented as an exhaustive list for requirements definition, but rather as a tool for examining and validating main architectural pathways through the system. This is in part due to a desire to present quality over quantity but also to recognize the requirements gathering already performed by the EVS team, validated by their own first-hand experience, and delivered in the “Consolidated server requirements” document. In many ways this provides the finer grain detail, but leaves us to ensure we are still seeing the “forest through the trees”.***

***NOTE: On the other end of the spectrum, this document will at times describe scenarios that exceed requirements for phase 1 deliverables. This is deemed beneficial in order to allow the definition of architectural underpinnings that do not preclude future improvements. The requirements document will act as final arbiter in defining what is in or out of scope for phase 1.***

## 1.3 Terminology

- For purposes of this document, ‘Vocabulary’ and ‘Terminology’ are interchangeable.

## 1.4 Related Documents

Description	Date / Version	Owner
LexBIG Software Requirements Specification	August 31, 2005/1.0	Mayo Foundation
LexBIG Architecture Document	<pending>	Mayo Foundation
LexBIG Administration Guide (Install and Config)	<pending>	Mayo Foundation
LexBIG Programmer Guide (API and connectivity)	<pending>	Mayo Foundation

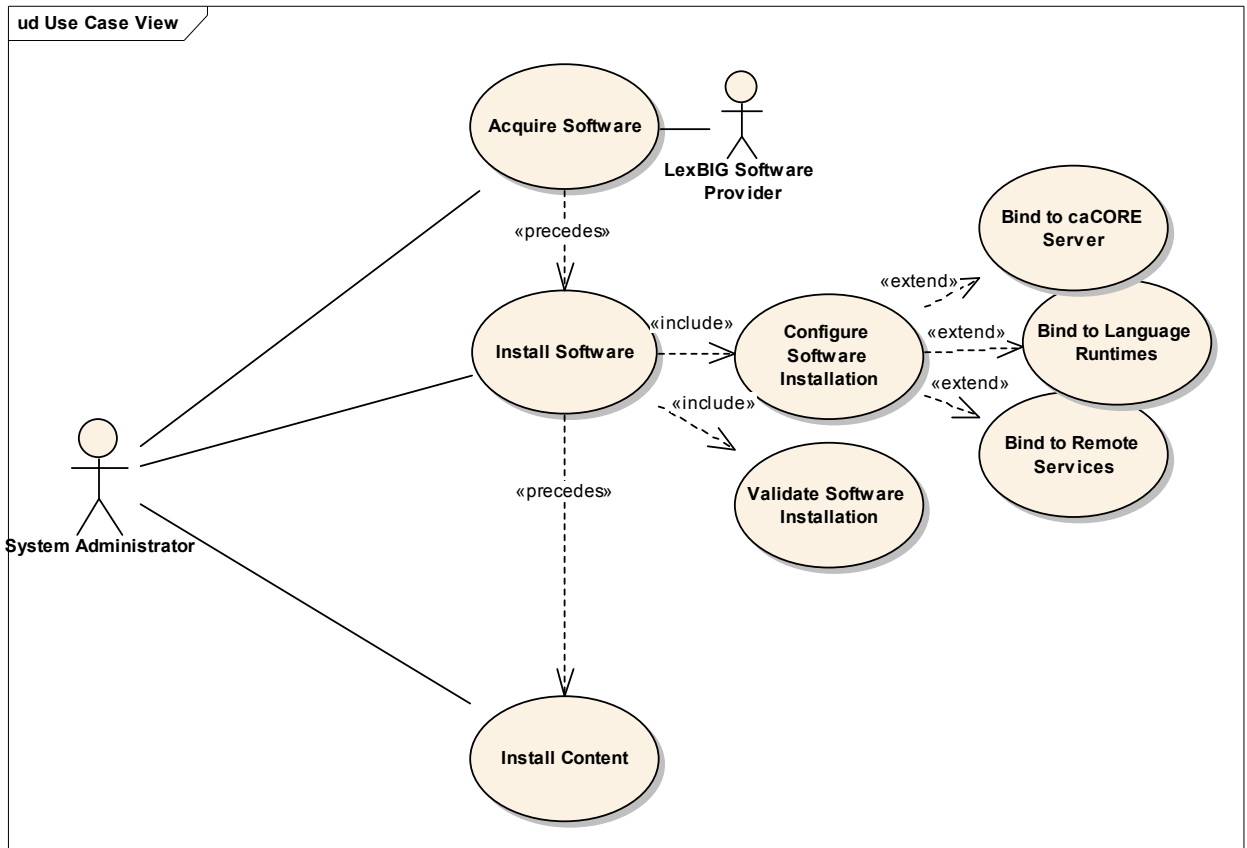
## 2 Actors

Name	Description
caBIG Modeler	A person responsible for creating data element models for a caBIG application
caTIES Annotator Program	Software that performs textual processing and annotation of pathology reports
caTIES Retrieval Application	Software application for retrieval of pathology reports
Client Application	A software application that utilizes LexBIG vocabulary services, either through direct discovery/invoke or via the caCORE EVS API.
End User	A person who directly or indirectly uses LexBIG services to accomplish a task.
LexBIG Server	Software that implements the LexBIG API(s)
LexBIG Software Provider	Organization responsible for maintaining the LexBIG vocabulary server software.
LexBIG User	A person or organization that uses LexBIG content in a modeling or development role
LexBIG Vocabulary Distributor	An organization that maintains a LexBIG server node on the grid. This server carries a registered copy of one or more LexBIG vocabularies.
NCI Thesaurus Browser	The existing NCI Thesaurus Browsing application
NCI MetaThesaurus Browser	The existing NCI MetaThesaurus Browsing Application
Research Investigator	A person active in clinical or basic science research.
System Administrator	The system administrator is responsible for installation, upgrades, and on-site monitoring of the vocabulary service software.
Vocabulary Publisher	A person or organization that is responsible for maintaining and publishing definitive revisions of vocabularies. Ideally, the entity that authors and maintains the vocabulary will also serve in the role of a primary publisher, but any individual or organization that reformats, integrates, or otherwise alters the format or content of a primary resource is also included as part of this role.
Vocabulary Server	The LexBIG runtime environment, alone or coupled with caCORE server.

## 3 Use Cases

### 3.1 Installation

This section describes use cases to the installation of the vocabulary server and vocabulary content. These use cases support the local and federation goals of the caBIG LexGrid project. Installation of server and content provide caBIG community members the ability to deploy a local instance of the vocabulary server and content. Local installation may be desired for performance, accessibility, and control reasons. Individual caBIG community members may choose to install/host a caBIG approved vocabulary and reference different sites for other vocabularies. Upon successful, installation application developers will be able to use the caCORE SDK which bind to LexGrid API or access LexGrid API directly to support vocabulary access and usage.



#### 3.1.1 Install Software

Install software provides the general use cases for acquiring, installing, configuring, and validating software installation. Multiple instances of a vocabulary server with the same software version or different software versions are both valid scenarios that individual caBIG members may configure.

##### 3.1.1.1 Acquire Software

Use Case ID	INS_UC_01
Primary Actor	System Administrator
Secondary Actors	LexBIG Software
Brief Description	Receive software installation package from application provider.
Pre-conditions	Internet connectivity

<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Administrator selects link to download software from provider web page.</li> <li>2) System provides option to register interest in future updates. **</li> <li>3) System provides access to license terms, requests confirmation if restrictive.</li> <li>4) Administrator confirms the download.</li> <li>5) System transmits software.</li> <li>6) System logs download date/time and other statistics.</li> </ol>
<b>Post Conditions</b>	Administrator has local copy of software installation package.
<b>Notes</b>	** Extent and nature of registration mechanism will be determined on completion of LexBIG architecture.

### 3.1.1.2 Install Software

<b>Use Case ID</b>	<b>INS_UC_02</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Actor installs and verifies the LexBIG runtime environment.
<b>Pre-conditions</b>	<p>Actor has obtained install package from application provider.</p> <p>Target system meets minimum hardware requirements (described in admin guide).</p> <p>Prerequisite software is installed and configured (described in admin guide).</p>
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor opens and steps through installation documentation/instructions.</li> <li>2) System provides guided installation using semi-automated programs/scripts to assist with file placement, configuration settings, etc.</li> </ol>
<b>Post Conditions</b>	Runtime environment is active and prepared for installation of specific vocabularies.
<b>Notes</b>	<p>Includes</p> <ul style="list-style-type: none"> <li>• <a href="#">Configure Software Installation</a> (INS_UC_03)</li> <li>• <a href="#">Validate Software Installation</a> (INS_UC_07)</li> </ul>

### 3.1.1.3 Configure Software Installation

<b>Use Case ID</b>	<b>INS_UC_03</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Actor customizes the LexBIG runtime installation based on preference and target environment.
<b>Pre-conditions</b>	See <a href="#">Install Software</a> (INS_UC_02) pre-conditions.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor proceeds with configuration steps defined in installation instructions.</li> <li>2) System prompts actor for repository configuration (e.g. database name and connectivity settings, etc)</li> <li>3) Actor provides required settings and indicates to proceed.</li> <li>4) System presents actor with option to bind the LexBIG runtime to a caCORE server installation to fulfill EVS API requests.</li> <li>5) Actor optionally proceeds with configuration as described by extension point <a href="#">Bind to caCORE Server</a> (INS_UC_04).</li> <li>6) System presents user with option to bind the LexBIG runtime to programming language runtime environments.</li> <li>7) Actor optionally proceeds with configuration as described by extension point <a href="#">Bind to Language Runtimes</a> (INS_UC_05).</li> <li>8) System presents user with option to bind the LexBIG runtime to remote-accessible services.</li> <li>9) Actor optionally proceeds with configuration as described by extension point <a href="#">Bind to Remote Services</a> (INS_UC_06).</li> <li>10) System centrally records all configuration settings, when handled programmatically, in anticipation of potential serviceability and uninstall needs.</li> </ol>
<b>Post Conditions</b>	LexBIG runtime is installed and configured, but not verified.
<b>Notes</b>	System options will be presented and carried out via documentation or a semi-automated install program.

## 3.1.1.4 Bind to caCORE Server

<b>Use Case ID</b>	<b>INS_UC_04</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Binds the LexBIG runtime to an existing installation of the caCORE server.
<b>Pre-conditions</b>	LexBIG runtime files are installed. A caCORE server is installed and properly configured on the target system.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor chooses to bind the LexBIG runtime to caCORE server.</li> <li>2) System requests location of caCORE server installation.</li> <li>3) Actor provides location and indicates to proceed.</li> <li>4) System modifies caCORE configuration as required to establish linkage (tbd) System takes any action necessary to cause changes to take effect (e.g. restart the server).</li> </ol>
<b>Post Conditions</b>	EVS API requests are fulfilled through the LexBIG runtime.
<b>Notes</b>	<p>Exact nature of linkage from caCORE server to LexBIG runtime services to be determined per completion of LexBIG architecture (pending).</p> <p>This use case is currently written assuming installation of the caCORE server and LexBIG runtime on the same machine. If required, remote access to the LexBIG runtime may also be considered and linked through remote services (see <a href="#">Bind to Remote Services</a>, INS_UC_06).</p>

## 3.1.1.5 Bind to Language Runtimes

<b>Use Case ID</b>	<b>INS_UC_05</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Optimizes access to the LexBIG runtime for programs deployed on the same system.
<b>Pre-conditions</b>	LexBIG runtime files are installed.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor chooses to bind the LexBIG runtime to language-specific environments.</li> <li>2) System presents actor with supported choices (e.g. Java)</li> <li>3) Actor makes selection and indicates to proceed.</li> <li>4) System prompts actor for any additional required information (language-dependent).</li> <li>5) Actor provides requested information, if any, and indicates to proceed.</li> <li>6) System makes respective changes (e.g. adding LexBIG code to Java system classpath)</li> </ol>
<b>Post Conditions</b>	LexBIG runtime API is directly accessible by the selected programming languages.
<b>Notes</b>	

## 3.1.1.6 Bind to Remote Services

<b>Use Case ID</b>	<b>INS_UC_06</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Registers services for subsequent discovery and invocation by client programs.
<b>Pre-conditions</b>	LexBIG runtime files are installed. Target service environments are active and accessible.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor chooses to bind the LexBIG runtime as services for remote access.</li> <li>2) System presents user with supported options (e.g. SOAP-based access through local web service container, registration against a centrally maintained vocabulary service index, registration as caGRID services, etc).</li> <li>3) Actor selects service option(s) as target(s) for installation.</li> <li>4) System prompts user for any required info (e.g. a web server port).</li> <li>5) Actor provides requested information and confirms installation.</li> <li>6) System carries out service configuration.</li> </ol>
<b>Post Conditions</b>	Services can be discovered and invoked by Java/J2EE, .NET client applications, etc.
<b>Notes</b>	Supported service architectures will be determined as architecture proceeds. The primary purpose of this use case is to provide for some type of direct service-oriented language-independent access to the LexBIG vocabulary services.

## 3.1.1.7 Validate Software Installation

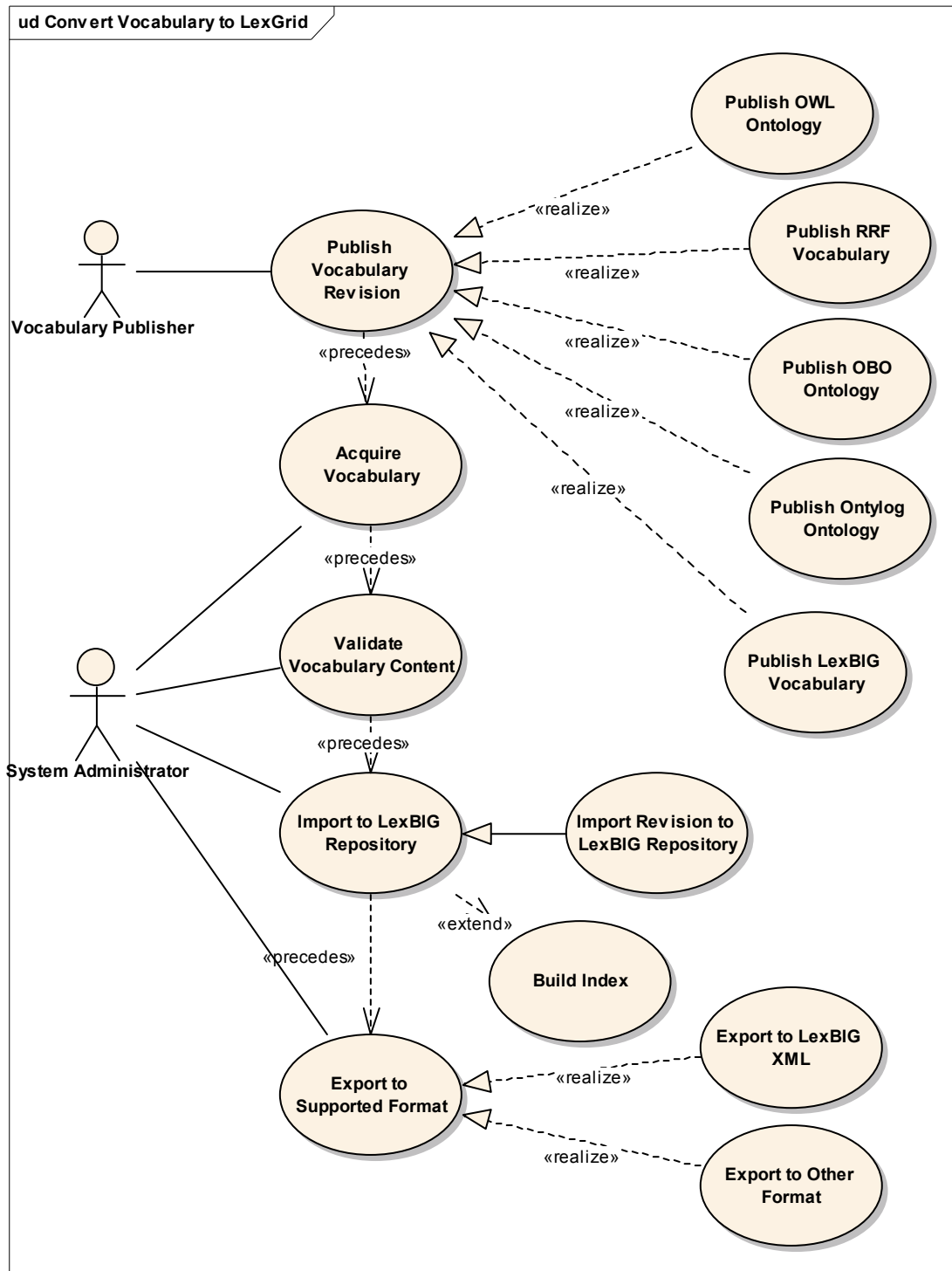
<b>Use Case ID</b>	<b>INS_UC_07</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Evaluates and reports installation status.
<b>Pre-conditions</b>	LexBIG runtime files are installed. Content repository is configured. Requested caCORE, language, and remote service bindings are configured.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor initiates validation script or program.</li> <li>2) System verifies content repository is accessible.</li> <li>3) System installs test vocabulary to the repository.</li> <li>4) System executes and evaluates result of build verification test suite. Tests are partitioned and run or ignored as appropriate based on availability of caCORE, direct language, and remote service bindings.</li> </ol>
<b>Post Conditions</b>	If validation succeeds, server is available for general use. Otherwise, required information is logged and reported for analysis and serviceability.
<b>Notes</b>	

## 3.1.1.8 Install Content

<b>Use Case ID</b>	<b>INS_UC_08</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Import and install LexBIG vocabulary content
<b>Pre-conditions</b>	
<b>Flow of Events</b>	Refer to <a href="#">Content Conversion and Load</a> (3.2) for detailed information.
<b>Post Conditions</b>	If validation succeeds, server is available for general use. Otherwise, required information is logged and reported for analysis and serviceability.
<b>Notes</b>	

### 3.2 Content Conversion and Load

Vocabulary publishers provide content in a variety of formats. To be used within the lexical grid, a vocabulary must be converted and stored in a format that is accessible through the standardized service software. Any updates or revisions must be recognized, converted and synchronized as well.



### 3.2.1 Publishing and Acquisition

This section describes process related to the distribution of published vocabularies from vocabulary owners (or redistributors) to the LexBIG host system.

#### 3.2.1.1 Publish Vocabulary Revision

<b>Use Case ID</b>	<b>VVC_UC_01</b>
<b>Primary Actor</b>	Vocabulary Publisher
<b>Secondary Actors</b>	System Administrator
<b>Brief Description</b>	Vocabulary publisher releases a new version of the vocabulary for distribution
<b>Pre-conditions</b>	
<b>Flow of Events</b>	1) Publisher places released vocabulary on a known site 2) Publisher notifies interested parties that the vocabulary is available and how to get it
<b>Post Conditions</b>	
<b>Notes</b>	1) Formats other than those supplied above will need to be converted by a Vocabulary Distributor before they can be made available within the grid. Some tools will be made available for this process 2) In many cases, vocabulary publishers will be completely outside the caBIG grid environment (e.g. CAP, WHO, ISO, etc.)

#### 3.2.1.2 Acquire Vocabulary

<b>Use Case ID</b>	<b>VVC_UC_02</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	Vocabulary Publisher
<b>Brief Description</b>	Administrator receives published vocabulary from vocabulary publisher
<b>Pre-conditions</b>	
<b>Flow of Events</b>	<provider-specific>
<b>Post Conditions</b>	Vocabulary files are available for validation on the target system
<b>Notes</b>	Supported formats include RRF, OWL-DL, and LexBIG XML.

#### 3.2.1.3 Validate Vocabulary Content

<b>Use Case ID</b>	<b>VVC_UC_03</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Verify the format and structure of the acquired vocabulary
<b>Pre-conditions</b>	
<b>Flow of Events</b>	Administrator uses LexBIG tooling to verify that the acquired vocabulary is complete, consistent and well formed
<b>Post Conditions</b>	Vocabulary files are available on the install system in a supported format.
<b>Notes</b>	Supported formats include RRF, OWL-DL, and LexBIG XML.

### 3.2.2 Import

This section describes process related to the transformation and integration of downloaded vocabularies to the LexBIG server repository to enable access by the LexBIG runtime environment.

#### 3.2.2.1 Import to LexBIG Repository

<b>Use Case ID</b>	<b>CVL_UC_01</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Installs a new vocabulary to the LexBIG repository for subsequent access via the LexBIG runtime API and services.
<b>Pre-conditions</b>	1. LexBIG import toolkit is installed 2. LexBIG runtime is installed and verified (see <a href="#">INS_UC_07</a> ) 3. Vocabulary is available in a format directly consumable by LexBIG import tools.
<b>Flow of Events</b>	1) Actor selects and runs the appropriate tool to load the vocabulary. 2) System requests information (e.g. location and format) for the vocabulary to install.

	3) Actor provides requested information and indicates to proceed. 4) System initiates the load and evaluates success or failure. 5) If successful, actor may initiate rebuild of affected lexical indices (see <a href="#">CVL_UC_04</a> ) 6) Actor runs validation/verification tests
<b>Post Conditions</b>	If the load succeeds, the vocabulary is available for access via the LexBIG runtime API and services. Otherwise, information pertaining to the failure is logged and reported for analysis and serviceability.
<b>Notes</b>	Structured

### 3.2.2.2 Import Revision to LexBIG Repository

<b>Use Case ID</b>	<b>CVL_UC_03</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Installs a new version of an already loaded vocabulary to a LexBIG server repository
<b>Pre-conditions</b>	
<b>Flow of Events</b>	1) Actor selects and runs the appropriate tool to update the vocabulary. 2) System presents choice of keeping or replacing previous version(s) of vocabulary. 3) Actor makes selection and indicates to proceed. 4) General steps mirroring a scratch <a href="#">Load</a> use case, ( <a href="#">CVL_UC_02</a> )
<b>Post Conditions</b>	If the load succeeds, the revision is available for access via the LexBIG runtime API and services. Otherwise, information pertaining to the failure is logged and reported for analysis and serviceability.
<b>Notes</b>	

### 3.2.2.3 Build Index

<b>Use Case Id</b>	<b>CVL_UC_04</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Actor creates LexBIG indexes for newly loaded data.
<b>Pre-conditions</b>	1) Actor has completed load of a compatible vocabulary or revision 2) LexBIG indexer tool is available 3) Optional – LVG installation is available
<b>Flow of Events</b>	1) Actor launches the LexBIG indexing tool. 2) System requests necessary parameters for indexing. 3) Actor provides requested information and indicates to proceed (e.g. address and authentication information for the LexBIG server, index output location). 4) Optional – Actor enters LVG configuration information if they wish to build a normalized index. 5) Actor launches the index process. 6) System validates the parameters and indexes the data. 7) Actor modifies the configuration for the LexBIG server so that it knows about the new index.
<b>Post Conditions</b>	Actor has constructed and registered an index on the LexBIG data with the LexBIG server.
<b>Notes</b>	

### 3.2.3 Export

This section describes process related to the extraction of vocabularies from the LexBIG repository to file formats supporting distribution or other programmatic access.

#### 3.2.3.1 Export to LexBIG XML

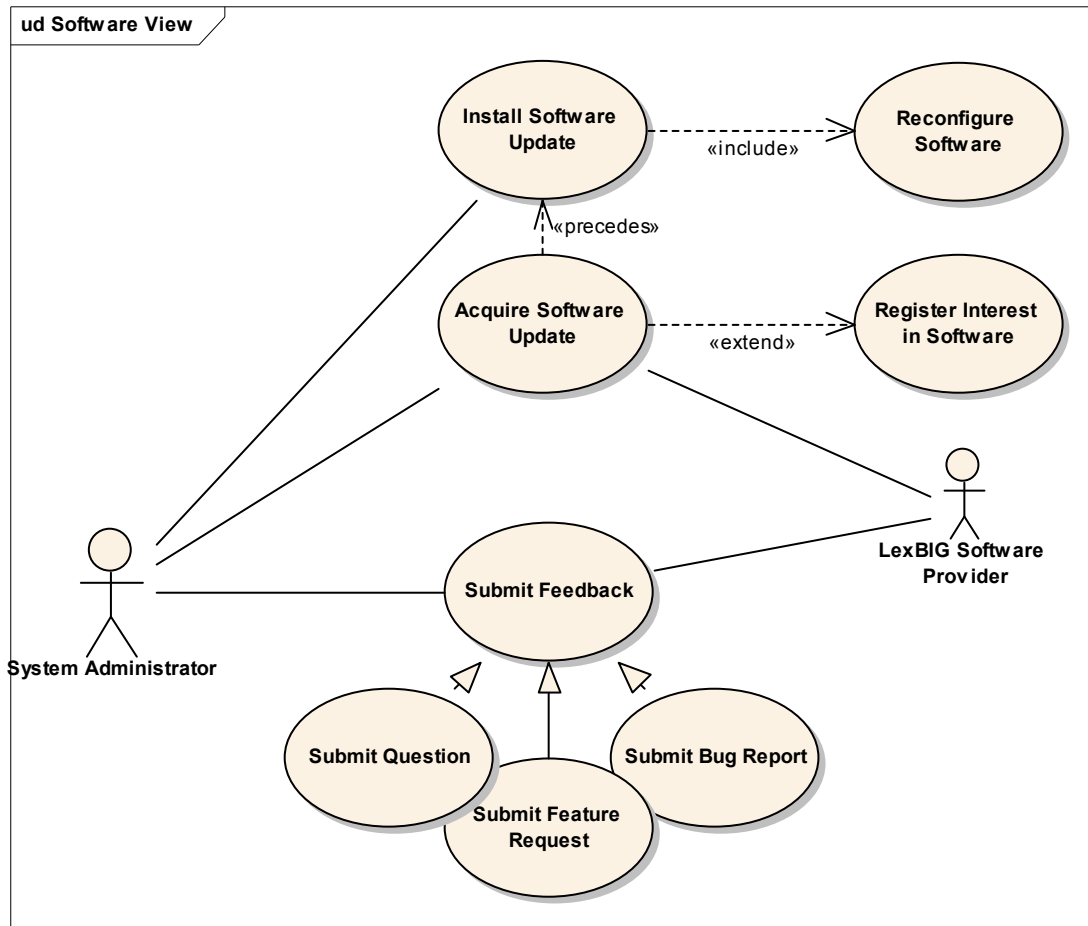
<b>Use Case ID</b>	<b>CVL UC 05</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Exports a LexBIG XML representation of a vocabulary from the LexBIG repository for analysis and subsequent processing by other programs (example of generalized export capability).
<b>Pre-conditions</b>	<ol style="list-style-type: none"> <li>1) LexBIG runtime is installed and verified (see <a href="#">INS UC 07</a>)</li> <li>2) Vocabulary in question exists in the LexBIG repository.</li> </ol>
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor selects and runs the appropriate tool to export the vocabulary.</li> <li>2) System requests information (e.g. location and identity) of the vocabulary to export.</li> <li>3) Actor provides requested information and indicates to proceed.</li> <li>4) System initiates the export and evaluates success or failure.</li> <li>5) If successful, LexBIG XML representation is written to designated target file.</li> </ol>
<b>Post Conditions</b>	If the export succeeds, the vocabulary is available in the converted format; runtime access to the vocabulary is not impacted. Otherwise, information pertaining to the failure is logged and reported for analysis and serviceability.
<b>Notes</b>	

### 3.3 Maintenance

This section describes update and serviceability of the vocabulary server and content. Vocabulary server software and vocabulary content requires active management and maintenance. Server software will need upgrades as functional enhancements, bug fixes, and caCORE binding changes occur. Vocabulary content will need to be periodically updated to reflect changes in domain knowledge. Frequency of vocabulary changes will vary depending on the caBIG member or vocabulary publisher. Despite the variability of content changes, the system will provide a mechanism to support these changes.

#### 3.3.1 Software Updates

Software update provides the general use cases for acquiring, installing, configuring, and validating new versions of the code base comprising the LexBIG runtime environment.



##### 3.3.1.1 Acquire Software Update

Use Case ID	UPS_UC_01
Primary Actor	System Administrator
Secondary Actors	LexBIG Software Provider
Brief Description	Administrator receives software installation package from application provider.
Pre-conditions	Internet Connectivity

<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Administrator receives notification of update availability from application provider (if previously registered); otherwise actor becomes aware of update through active monitoring of provider site or other communication.</li> <li>2) System provides ability to register interest in future updates ** (previously registered parties can modify choice or contact information)</li> <li>3) Actor provides requested information or bypasses registration.</li> <li>4) System provides access to license terms, requests confirmation if restrictive.</li> <li>5) Actor confirms the download.</li> <li>6) System transmits software.</li> <li>7) System logs download date/time and other statistics.</li> </ol>
<b>Post Conditions</b>	Actor has local copy of software update package.
<b>Notes</b>	** Extent and nature of registration mechanism will be determined on completion of LexBIG architecture.

### 3.3.1.2 Install Software Update

<b>Use Case ID</b>	<b>UPS_UC_02</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Actor installs the updated code and reinitializes the LexBIG runtime environment.
<b>Pre-conditions</b>	Actor has obtained update package from application provider.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor opens and steps through update documentation/instructions.</li> <li>2) System also provides guided installation using semi-automated programs/scripts to assist with file placement, configuration settings, etc.</li> <li>3) System restarts any impacted servers and revalidates the installation (see <a href="#">Validate Software Installation</a>, INS_UC_07). User is notified of operation success or failure. On success, system performs cleanup of temporary or unused files. On failure, applicable information is logged and reported for analysis and serviceability; actor is presented options to revert to pre-update code base.</li> </ol>
<b>Post Conditions</b>	Runtime environment is active and prepared for incoming requests.
<b>Notes</b>	Includes <a href="#">Reconfigure Software</a> (UPS_UC_03)

### 3.3.1.3 Reconfigure Software

<b>Use Case ID</b>	<b>UPS_UC_03</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	
<b>Brief Description</b>	Actor customizes the LexBIG runtime installation on change to code base.
<b>Pre-conditions</b>	Files for updated code base are installed.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor initiates update.</li> <li>2) System determines whether existing configuration settings are affected. If so, system revisits applicable steps of <a href="#">Configure Software Installation</a> (INS_UC_03).</li> </ol>
<b>Post Conditions</b>	LexBIG runtime is installed and configured, but not verified.
<b>Notes</b>	

### 3.3.1.4 Submit Feedback

<b>Use Case ID</b>	<b>UPS_UC_04</b>
<b>Primary Actor</b>	System Administrator
<b>Secondary Actors</b>	LexBIG Software Provider
<b>Brief Description</b>	Actor is provided a mechanism for ongoing dialog and issue resolution with the application provider.
<b>Pre-conditions</b>	Application provider has established an external presence on the internet and installed tools required to service requests from the caBIG community.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor selects service tool from link on provider web site.</li> <li>2) System interaction driven by web interface, specific to tools used (e.g. GForge)</li> <li>3) Actor completes submission of question, bug report, or feature request.</li> <li>4) System provides acknowledgement and estimated timeframe for response.</li> </ol>
<b>Post Conditions</b>	Feedback delivered.
<b>Notes</b>	

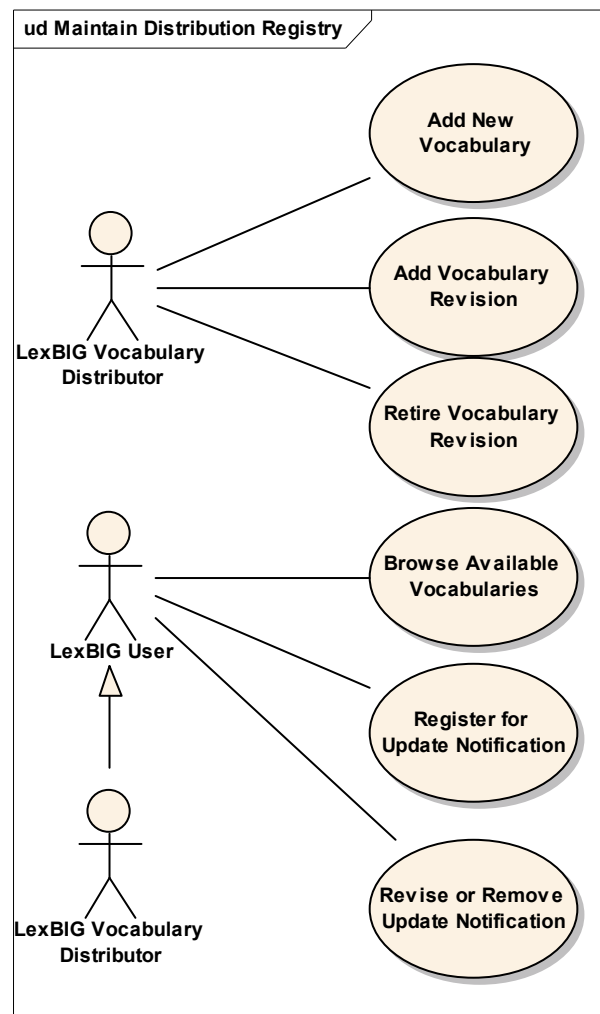
### 3.3.2 Maintain Vocabulary Registry

Vocabulary distribution is facilitated by the availability of centralized information regarding what vocabularies are available, what format they are available in, and where they may be acquired. In LexBIG, vocabulary registration and lookup is intended to address the following:

- **Registration of Availability and Interest:** Enables vocabulary providers to advertise new versions of a vocabulary and respective format(s) (e.g. OWL-DL, RRF, etc). Vocabulary consumers are provided the ability to monitor additions or request notification of changes. However, notification may not imply immediate access in LexBIG runtime environments. For example, the authors may indicate that ‘Bob’s terminology version 2.0 is now available in OWL-DL’, which by itself does not imply availability as a node on the lexical grid.
- **Registration of Services:** Enables programs to access a central site to locate a node or service provider for a vocabulary on the lexical grid. The program is able to use this information to resolve programmatic access to vocabulary content and metadata via LexBIG services (e.g. for concept lookup and cross-terminology mapping). This registration is sometimes referred to as the ‘LexBIG Service Index’.

While all the above functions might ultimately be satisfied through a single combined registry, the expectation in Phase 1 is that registration of availability and interest will be satisfied through some type of commodity interface (e.g. e-mail distribution lists). Registration of LexBIG nodes and services will receive comparatively higher priority in order to satisfy functional requirements.

**NOTE:** This section describes functions for registration of availability and interest, which are not considered in scope of LexBIG phase 1 deliverables. Registration of services, if required, will be addressed by architecture supporting the [Content Conversion and Load](#) use cases.



## 3.3.2.1 Add New Vocabulary

<b>Use Case ID</b>	<b>REG UC 01</b>
<b>Primary Actor</b>	LexBIG Vocabulary Distributor
<b>Secondary Actors</b>	
<b>Brief Description</b>	Register availability of a new vocabulary in the registry
<b>Pre-conditions</b>	New vocabulary is formatted and available for use on external web site
<b>Flow of Events</b>	1) Distributor records description of vocabulary in registry 2) Distributor records format, download location in registry
<b>Post Conditions</b>	Vocabulary is registered as available for external use and upload
<b>Notes</b>	Registry assumes the existence of pre-established “canonical” publication formats that can be automatically parsed and unambiguously interpreted by consuming organizations. As an example, “OWL XML” is not sufficient – an exact format and semantics must be established in advance for publication in this environment.

## 3.3.2.2 Add Vocabulary Revision

<b>Use Case ID</b>	<b>REG UC 02</b>
<b>Primary Actor</b>	LexBIG Vocabulary Distributor
<b>Secondary Actors</b>	
<b>Brief Description</b>	Register a new version of an existing vocabulary
<b>Pre-conditions</b>	A new version of a vocabulary is available for use on external web site
<b>Flow of Events</b>	1) Distributor records description of vocabulary revision into registry 2) Distributor records format, download location in registry 3) Distributor records format and location of formal change description in registry (where applicable) 4) Notifications are sent to all Vocabulary Users who have registered an interest in this vocabulary.
<b>Post Conditions</b>	Vocabulary and change set are available for use and upload
<b>Notes</b>	Change record formats need to be in pre-established semantics and formats. These can vary from opaque blobs to (ideally) machine interpretable update records.

## 3.3.2.3 Retire Vocabulary Revision

<b>Use Case ID</b>	<b>REG UC 03</b>
<b>Primary Actor</b>	LexBIG Vocabulary Distributor
<b>Secondary Actors</b>	
<b>Brief Description</b>	Record the fact that a version or versions of a vocabulary are no longer available for redistribution.
<b>Pre-conditions</b>	Vocabulary versions to be retired are recorded as active in the registry Vocabulary versions are available on an for use on an external web site
<b>Flow of Events</b>	1. Distributor records the fact that the revision is no longer available 2. Distributor records possible secondary locations where archives of the version may be located 3. Notifications are sent to all Vocabulary Redistributors who have registered an interest in this vocabulary.
<b>Post Conditions</b>	Vocabulary version is marked as no longer available. Vocabulary version may be removed from external web site.
<b>Notes</b>	There also needs to be a use case for the complete removal of a vocabulary, which will include the removal of all notification requests from the registry as well.

## 3.3.2.4 Browse Available Vocabularies

<b>Use Case ID</b>	<b>REG_UC_04</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	Browse the vocabulary registry for documentation on published vocabularies, versions and formats.
<b>Pre-conditions</b>	User is authorized to view registry
<b>Flow of Events</b>	1) User uses a combination of search and browse techniques to peruse the vocabularies that are available for upload and use on the grid
<b>Post Conditions</b>	
<b>Notes</b>	Behavior of this use case needs to be refined at some point

## 3.3.2.5 Register for Update Notification

<b>Use Case ID</b>	<b>REG_UC_05</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	Register to be notified whenever the state of a vocabulary is updated in the registry
<b>Pre-conditions</b>	User is authorized to register for notification
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) User selects a list of vocabularies for which notification is needed</li> <li>2) User selects a notification mechanism (RSS, e-mail, etc.) from a list of mechanisms available on the register</li> <li>3) User enters appropriate contact information for the selected mechanism</li> <li>4) User enters secondary contact information (typically e-mail) for out-of-band communication.</li> <li>5) An initial notification is sent to the redistributors for each of the registered vocabularies</li> <li>6) Where appropriate, redistributors responds to confirm receipt of initial notification message</li> </ol>
<b>Post Conditions</b>	User is registered to receive update notifications for selected vocabularies
<b>Notes</b>	Subsequent notifications do not require a confirmation. Where appropriate, however, negative feedback on the channel (unable to deliver message, unable to connect), should result in attempts to retransmit and/or the placement of a temporary hold on notifications until connection problem is corrected.

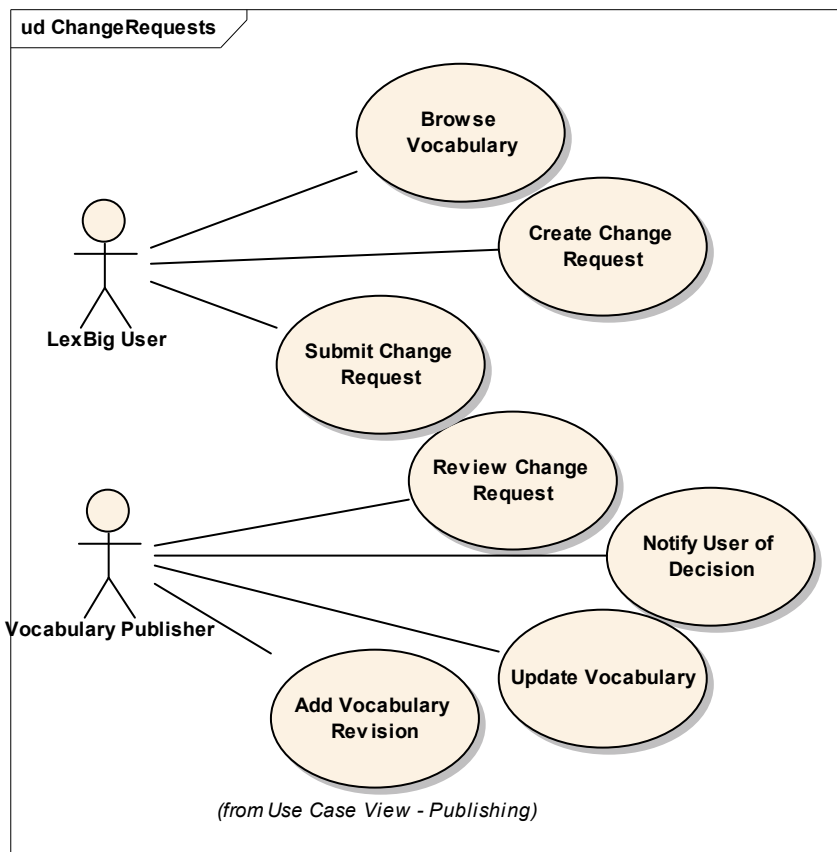
## 3.3.2.6 Revise or Remove Update Notification

<b>Use Case ID</b>	<b>REG_UC_06</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	Revise or remove a notification entry for a particular vocabulary
<b>Pre-conditions</b>	User or appropriate proxy (system administrator, etc) are authorized to access registry
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) User is presented list of their current notification requests</li> <li>2) User selects one or more entries from the list</li> <li>3) User changes appropriate notification information and/or requests to be removed from notification list</li> <li>4) If primary contact information is changed, an initial notification is sent over the new primary channel and confirmation is expected where appropriate</li> <li>5) If distributor is to be removed from one or more lists, a final notification event is sent via both primary and secondary channels</li> <li>6) All other changes are transmitted via the secondary notification channel. Verification is expected to complete the transaction where permitted by the medium</li> </ol>
<b>Post Conditions</b>	Notification records are updated appropriately
<b>Notes</b>	Subsequent notifications do not require a confirmation. Where appropriate, however, negative feedback on the channel (unable to deliver message, unable to connect), should result in attempts to retransmit and/or the placement of a temporary hold on notifications until connection problem is corrected.

### 3.3.3 Content Change Requests

The LexBIG User is presented with a uniform view of all LexBIG content, so all change requests are created in terms of LexBIG structure and semantics. Using the LexBIG tooling and model, the LexBIG user makes local changes that reflect the way that the user believes that the changed vocabulary should look. The change request tooling guides the user through the process by prompting for required fields, links, etc.

**NOTE:** The ‘Browse Vocabulary’ and ‘Add Vocabulary Revision’ items are included in the diagram below to provide additional context and clarity. Items specifically related to the submission and processing of change requests are not considered in scope of LexBIG Phase 1 deliverables.



#### 3.3.3.1 Create Change Request

<b>Use Case ID</b>	<b>CCR UC 01</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	(See <a href="#">Create Change Request Use Case</a> )
<b>Pre-conditions</b>	
<b>Flow of Events</b>	User creates the appropriate change request
<b>Post Conditions</b>	Change request is ready for submission
<b>Notes</b>	

## 3.3.3.2 Submit Change Request

<b>Use Case ID</b>	<b>CCR UC 02</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	User submits a change request package to the appropriate curator
<b>Pre-conditions</b>	User has created the necessary set of formatted change requests
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) User enters description of change, urgency and secondary contact information</li> <li>2) Change request package uses vocabulary registry to determine where and how to route the request</li> <li>3) User is notified that request has been submitted</li> </ol>
<b>Post Conditions</b>	Change request has been transmitted to the appropriate target vocabulary curator
<b>Notes</b>	

## 3.3.3.3 Review Change Request

<b>Use Case ID</b>	<b>CCR UC 03</b>
<b>Primary Actor</b>	Vocabulary Publisher
<b>Secondary Actors</b>	LexBIG User
<b>Brief Description</b>	Review a change request
<b>Pre-conditions</b>	Change request has been submitted by a LexBIG User
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Publisher receives change request</li> <li>2) Submitter is notified that request has been received</li> <li>3) Publisher reviews request and translates it into the semantics appropriate for the native vocabulary form</li> <li>4) Publisher interacts with the submitter as necessary to clarify the intent of the request</li> </ol>
<b>Post Conditions</b>	Vocabulary Publisher has a clear understanding of the intent and purpose of the request
<b>Notes</b>	

## 3.3.3.4 Notify User of Decision

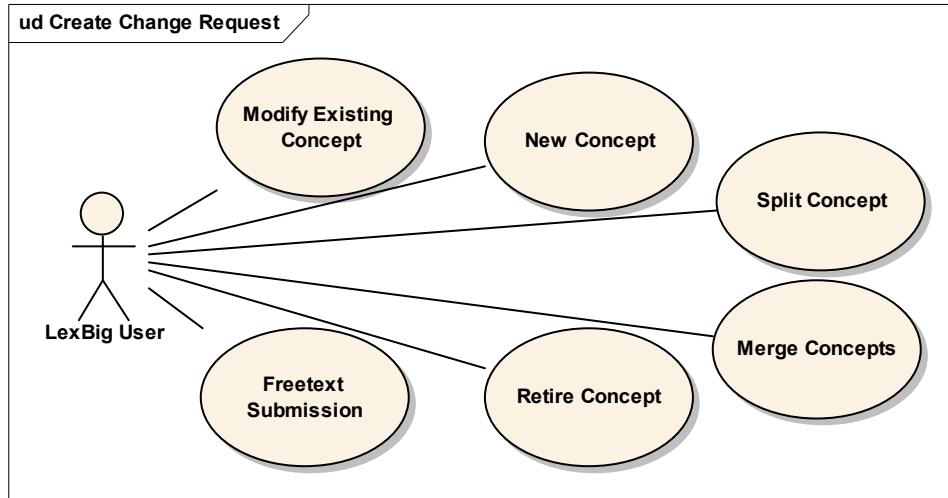
<b>Use Case ID</b>	<b>CCR UC 04</b>
<b>Primary Actor</b>	Vocabulary Publisher
<b>Secondary Actors</b>	LexBIG User
<b>Brief Description</b>	Determine the validity and time frame for a change request and notify the requesting user
<b>Pre-conditions</b>	Change request is clearly understood by curator
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1. Publisher determines whether request will be implemented, how it will be implemented and the timeframe for the release</li> <li>2. Publisher discusses any variations from user request with user</li> <li>3. Publisher sends requesting user a formal notification of the decision details, timing, etc.</li> </ol>
<b>Post Conditions</b>	User and Publisher have agreement on what is to occur
<b>Notes</b>	

## 3.3.3.5 Update Vocabulary

<b>Use Case ID</b>	<b>CCR UC 05</b>
<b>Primary Actor</b>	Vocabulary Publisher
<b>Secondary Actors</b>	
<b>Brief Description</b>	Publisher implements requested change in source vocabulary
<b>Pre-conditions</b>	User and curator have agreement on what is to occur
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1. Publisher makes appropriate updates to target vocabulary</li> <li>2. Updates are classified, validated, reviewed, etc.</li> <li>3. Revisions are converted to LexBIG form, reviewed and compared to user expectations</li> <li>4. Change records are created and documentation produced</li> </ol>
<b>Post Conditions</b>	Revised vocabulary is ready for publication
<b>Notes</b>	

### 3.3.4 Create Change Request (expanded)

This section describes specific sub-tasks to be considered when creating a change request conforming to the structure and semantics set forth by the LexBIG model (refer to test case [CCR\\_UC\\_01](#) for additional context).



#### 3.3.4.1 Modify Existing Concept

<b>Use Case ID</b>	<b>CCR_UC_01</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	Make changes to attributes of an existing concept
<b>Pre-conditions</b>	Concept to be modified has been selected.
<b>Flow of Events</b>	1) User adds or removes properties, definitions, synonyms, roles, etc. on the selected properties 2) Changes are recorded as a series of “delta” records
<b>Post Conditions</b>	“Delta” records, when applied to the existing concept will result in a state <i>in the LexGrid</i> model that represents the desired result.
<b>Notes</b>	Role changes impact two or more concepts, so the target concepts will be included in the change set as well. It is quite possible that the suggested role changes will not work, but there will be no way to tell for sure until the suggested changes are converted into the native format (e.g. OWL or Ontylog) and are run through a formal classifier. As such, these are change suggestions only.

#### 3.3.4.2 Create a New Concept

<b>Use Case ID</b>	<b>CCR_UC_02</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	User creates a new concept as a subtype of an existing concept.
<b>Pre-conditions</b>	Supertype concept has been selected
<b>Flow of Events</b>	1) User is presented with a LexBIG template 2) User enters presentations, definitions, etc. that represent the new concept 3) User uses browser to assign any additional roles, etc. that are needed
<b>Post Conditions</b>	“Delta” records, when applied to the existing concept will result in a state <i>in the LexGrid</i> model that represents the desired result.
<b>Notes</b>	(see: CCR_UC_01)

## 3.3.4.3 Split a Concept

<b>Use Case ID</b>	<b>CCR_UC_03</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	User requests that a concept be split into two new concepts
<b>Pre-conditions</b>	Concept to be split has been selected
<b>Flow of Events</b>	
<b>Post Conditions</b>	“Delta” records, when applied to the existing concept will result in a state <i>in the LexGrid</i> model that represents the desired result.
<b>Notes</b>	(see: CCR_UC_01)

## 3.3.4.4 Merge two Concepts

<b>Use Case ID</b>	<b>CCR_UC_04</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	User requests that two concepts be combined into one
<b>Pre-conditions</b>	Concepts to be merged have been selected
<b>Flow of Events</b>	
<b>Post Conditions</b>	“Delta” records, when applied to the existing concept will result in a state <i>in the LexGrid</i> model that represents the desired result.
<b>Notes</b>	(see: CCR_UC_01)

## 3.3.4.5 Retire a Concept

<b>Use Case ID</b>	<b>CCR_UC_05</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	User requests that a concept code be changed from “active” to “retired”
<b>Pre-conditions</b>	Concepts to be retired is selected
<b>Flow of Events</b>	
<b>Post Conditions</b>	“Delta” records, when applied to the existing concept will result in a state <i>in the LexGrid</i> model that represents the desired result.
<b>Notes</b>	(see: CCR_UC_01)

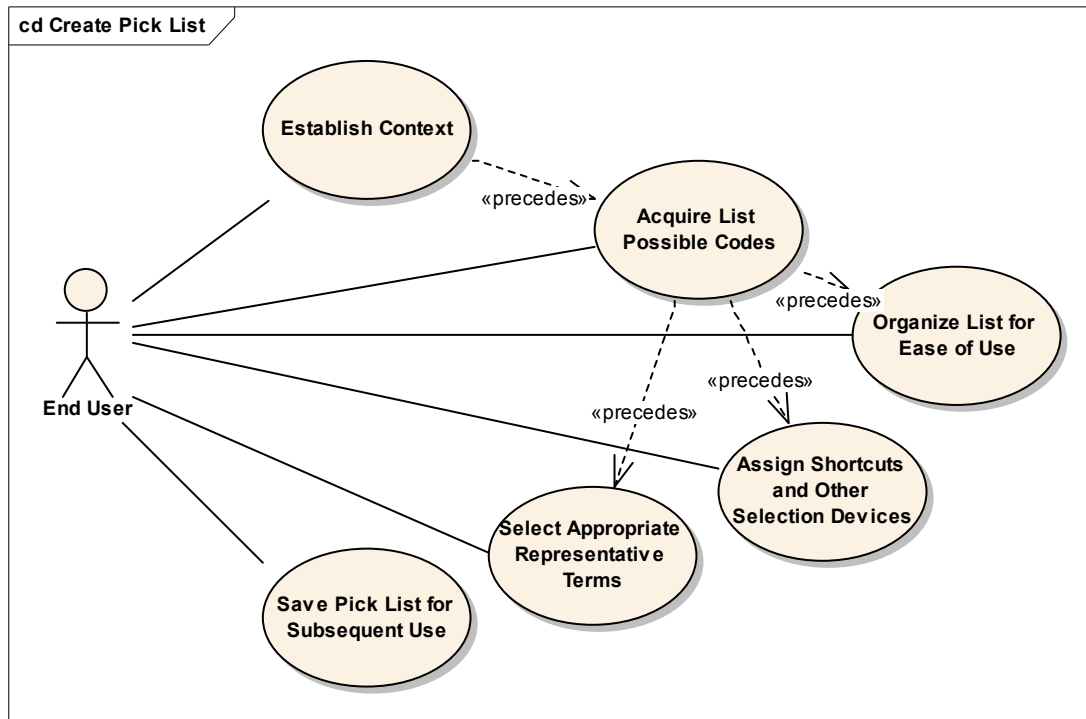
## 3.3.4.6 Free text Submission

<b>Use Case ID</b>	<b>CCR_UC_06</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	
<b>Brief Description</b>	User submits a textual description of the set of changes that is desired
<b>Pre-conditions</b>	Concept to be changed has been selected
<b>Flow of Events</b>	User enters text that describes the change or changes desired
<b>Post Conditions</b>	Target concept, along with change verbiage is included as part of the change record
<b>Notes</b>	(see: CCR_UC_01)

### 3.3.5 Create Pick List

An application user organizes and labels a list of concept codes that will be use for selecting values from one or more data entry fields. While codes may not be omitted from the list, they can be re-ordered, duplicated, grouped into hierarchies and assigned the appropriate labels and, where appropriate, shortcut mechanisms for efficiency and ease of use.

**NOTE:** Function specific to the creation and maintenance of pick lists is not considered in scope of LexBIG Phase 1 deliverables.



#### 3.3.5.1 Establish Context

Use Case ID	CPL_UC_01
Primary Actor	End User
Brief Description	The user records the circumstances in which the pick list applies
Pre-conditions	
Flow of Events	<ol style="list-style-type: none"> <li>1) A default context is acquired from the service session record</li> <li>2) The user supplies <ul style="list-style-type: none"> <li>• the application(s) in which this selection is applicable</li> <li>• the institutions(s), facilities, etc. where the selection applies</li> <li>• the individual user(s) or groups of users to which the selection applies</li> <li>• the desired language of the terms</li> </ul> </li> </ol>
Post Conditions	Context is established
Notes	

#### 3.3.5.2 Acquire List of Possible Codes

Use Case ID	CPL_UC_02
Primary Actor	End User
Brief Description	The user selects an attribute or value domain and gets a list of enumerated values
Pre-conditions	
Flow of Events	<ol style="list-style-type: none"> <li>1) User selects an attribute or enumerated value domain</li> <li>2) Vocabulary server returns a list of possible codes and associated designations for the domain</li> </ol>
Post Conditions	

<b>Notes</b>	
--------------	--

### 3.3.5.3 Organize list for ease of use

<b>Use Case ID</b>	<b>CPL_UC_03</b>
<b>Primary Actor</b>	End User
<b>Brief Description</b>	Restructure and reorder a list for maximum flexibility
<b>Pre-conditions</b>	Complete of possible values has been selected
<b>Flow of Events</b>	User reorders nodes, duplicates nodes, flattens hierarchies, establishes new non-selectable intermediate nodes, etc. to create a list that is optimal for quick selection
<b>Post Conditions</b>	List has a new hierarchy and/or order.
<b>Notes</b>	Vocabulary suppliers may object to the ability to restructure a list hierarchy because it adds unintended or invalid semantics to the resulting code set. If this is the case, restructuring should be discouraged or disallowed.

### 3.3.5.4 Select Appropriate Representative Terms

<b>Use Case ID</b>	<b>CPL_UC_04</b>
<b>Primary Actor</b>	End User
<b>Brief Description</b>	Assign “meaningful” representative terms to list nodes as needed
<b>Pre-conditions</b>	
<b>Flow of Events</b>	1) User selects an appropriate synonym from a list of possible designations of a concept code. 2) User creates a new representative term where appropriate
<b>Post Conditions</b>	
<b>Notes</b>	Vocabulary suppliers and regulatory agencies may object to the ability to assign new representative terms that aren’t a part of the actual vocabulary. When this is the case, the ability to create new representative terms should be restricted or disabled.

### 3.3.5.5 Assign Shortcuts and other Selection Devices

<b>Use Case ID</b>	<b>CPL_UC_05</b>
<b>Primary Actor</b>	End User
<b>Brief Description</b>	User assigns shortcuts, mnemonics and other access codes to list entries
<b>Pre-conditions</b>	
<b>Flow of Events</b>	
<b>Post Conditions</b>	
<b>Notes</b>	While this is an important step in application building, it tends to be platform and application specific and, as a consequence, may be out of scope.

### 3.3.5.6 Save Picklist for Subsequent Use

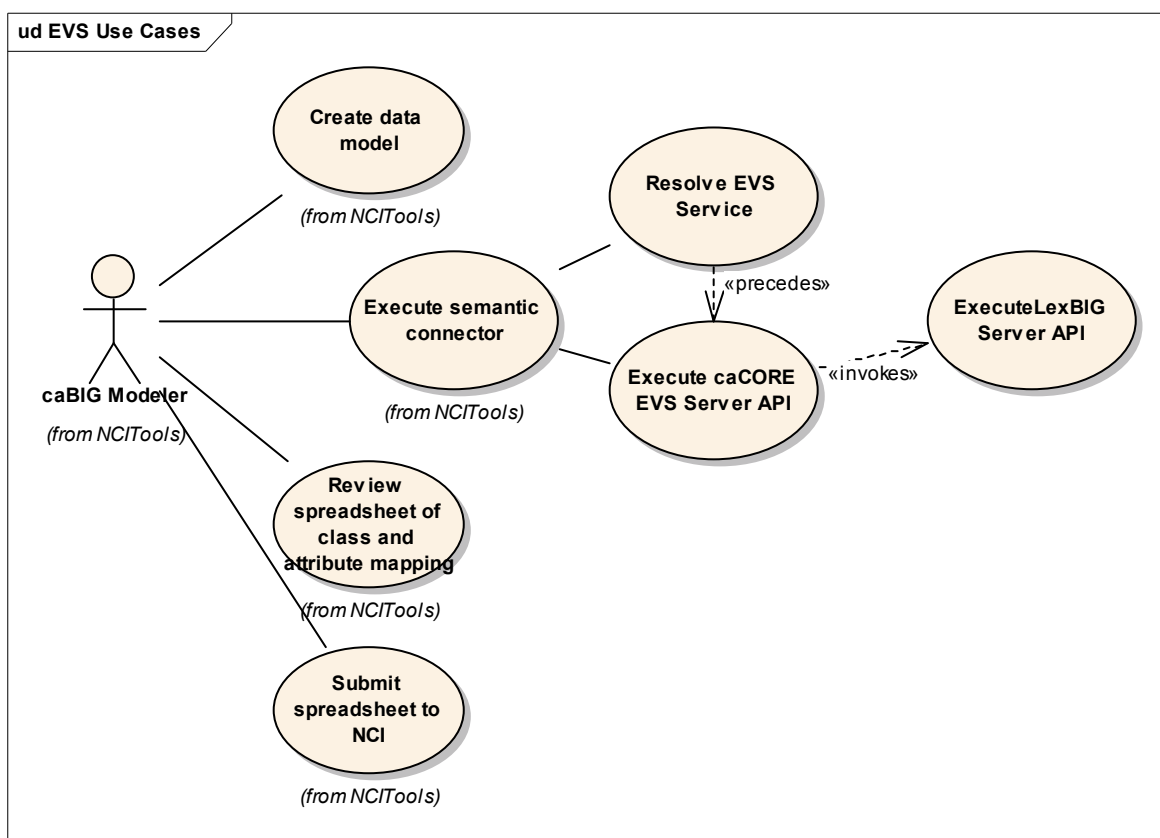
<b>Use Case ID</b>	<b>CPL_UC_06</b>
<b>Primary Actor</b>	Application User
<b>Brief Description</b>	User saves resulting list in repository for subsequent use
<b>Pre-conditions</b>	
<b>Flow of Events</b>	
<b>Post Conditions</b>	
<b>Notes</b>	Distribution, naming, scope, specialization, coordination w/ vocabulary changes, etc. are all unresolved issues at this point.

### 3.4 Runtime Access

This section describes typical runtime use cases for the vocabulary server. Runtime access includes access to the server through existing NCI tools such as NCI Browser or caCORE semantic connector. Runtime access may connect through caCORE or the LexBIG API.

#### 3.4.1 EVS Access: Annotate a Data Model using Semantic Connector

A modeler (user) creates a data model representing the data elements for their application. The data model is exported using XML Metadata Interchange (XMI). The modeler executes the semantic connector software. The semantic connector uses the XMI file and attempts to annotate the UML class and attribute names to the NCI thesaurus vocabulary. A spreadsheet is created enumerating the UML class, UML entity, concept code, concept name, and concept definition. If a match is not found, the concept information is blank. The modeler can refine the data model based on annotations and rerun the semantic connector. The final spreadsheet is submitted to NCI for review.



##### 3.4.1.1 Create data model

Use Case ID	EVS_UC_01
Primary Actor	caBIG Modeler
Secondary Actors	Modeling Tool – e.g. Enterprise Architect
Brief Description	Actor creates a data model based on the data requirements of the application or system.
Pre-conditions	Modeling tool that can generate XMI representation
Flow of Events	1) Create class (entities) e.g. Gene 2) Create attributes (columns) e.g. Identifier 3) Create necessary relationships 4) Review model for proper normalization (3NF) 5) Save model in XML Metadata Interchange format
Post Conditions	A model saved as XMI
Notes	

## 3.4.1.2 Execute semantic connector

<b>Use Case ID</b>	<b>EVS_UC_02</b>
<b>Primary Actor</b>	caBIG Modeler
<b>Secondary Actors</b>	
<b>Brief Description</b>	Actor runs the semantic connector using the XMI file as input
<b>Pre-conditions</b>	1) XMI file is created 2) Installation of caCORE SDK 3) Internet access
<b>Flow of Events</b>	1) XMI file is created 2) Copy XMI file into appropriate directory 3) Run semantic connector 4) Check acknowledgement of run
<b>Post Conditions</b>	A spreadsheet of UML class and attributes with concept mapping
<b>Notes</b>	

## 3.4.1.3 Resolve EVS Service

<b>Use Case Id</b>	<b>EVS_UC_03</b>
<b>Primary Actor</b>	
<b>Secondary Actors</b>	
<b>Brief Description</b>	Find and connect to a caCORE server.
<b>Pre-conditions</b>	1) caCORE has been installed and configured with a LexBIG EVS backend. 2) caCORE connection address has been published. 3) Actor has downloaded and installed the appropriate caCORE client application. 4) Actor has documentation for the caCORE client application.
<b>Flow of Events</b>	1) Client finds the caCORE connection address. 2) Client enters the connection address into the caCORE client as documented by the caCORE documentation. 3) Client executes method as specified by the caCORE documentation to connect to the caCORE server. 4) caCORE client attempts to make connection to the caCORE server. a. Success – caCORE client is ready for EVS calls. b. Failure – caCORE client returns an error message to the Actor.
<b>Post Conditions</b>	Connection established to a caCORE server.
<b>Notes</b>	

## 3.4.1.4 Invoke caCORE EVS API

<b>Use Case Id</b>	<b>EVS_UC_04</b>
<b>Primary Actor</b>	
<b>Secondary Actors</b>	
<b>Brief Description</b>	Client calls an EVS method from the caCORE API.
<b>Pre-conditions</b>	Client has completed EVS_UC_01
<b>Flow of Events</b>	1) Client chooses a method to call from the EVS caCORE API. 2) Client supplies requested parameters and executes the chosen method. 3) caCORE Client forwards parameters and method call to caCORE server. 4) caCORE Server forwards the parameters and method call to an appropriate LexBIG method call. 5) LexBIG method call returns the result(s) or error(s) to the caCORE server. 6) caCORE server returns result(s) or error(s) to the caCORE client. 7) The caCORE client returns the result(s) or error(s) to the Actor.
<b>Post Conditions</b>	Client has constructed and registered an index on the LexBIG data with the LexBIG server.
<b>Notes</b>	It is assumed that the LexBIG EVS caCORE server implements all of the methods as specified by the caCORE API that are implemented in the current caCORE server. It is assumed that the input parameters and results from the LexBIG EVS caCORE server will be the same as they are in the current caCORE implementation.

## 3.4.1.5 Review spreadsheet of class and attribute mapping

<b>Use Case ID</b>	<b>NCI UC 05</b>
<b>Primary Actor</b>	caBIG Modeler
<b>Secondary Actors</b>	
<b>Brief Description</b>	Actor opens spreadsheet file and reviews mapping of class names and attributes. Multiple mappings require the modeler to choose one of the choices. Blank mapping requires the modeler to review class/attribute name for spelling and camel case.
<b>Pre-conditions</b>	Spreadsheet software capable of reading comma separated file format
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Open annotated file</li> <li>2) Review attributes for multiple mappings or missing values</li> <li>3) Review class/attribute naming for missing values</li> <li>4) Refine model and repeat semantic connector run</li> <li>5) For missing or incorrect mappings enter appropriate concept definition</li> <li>6) Submit to NCI</li> </ol>
<b>Post Conditions</b>	A reviewed spreadsheet file with appropriate comments
<b>Notes</b>	

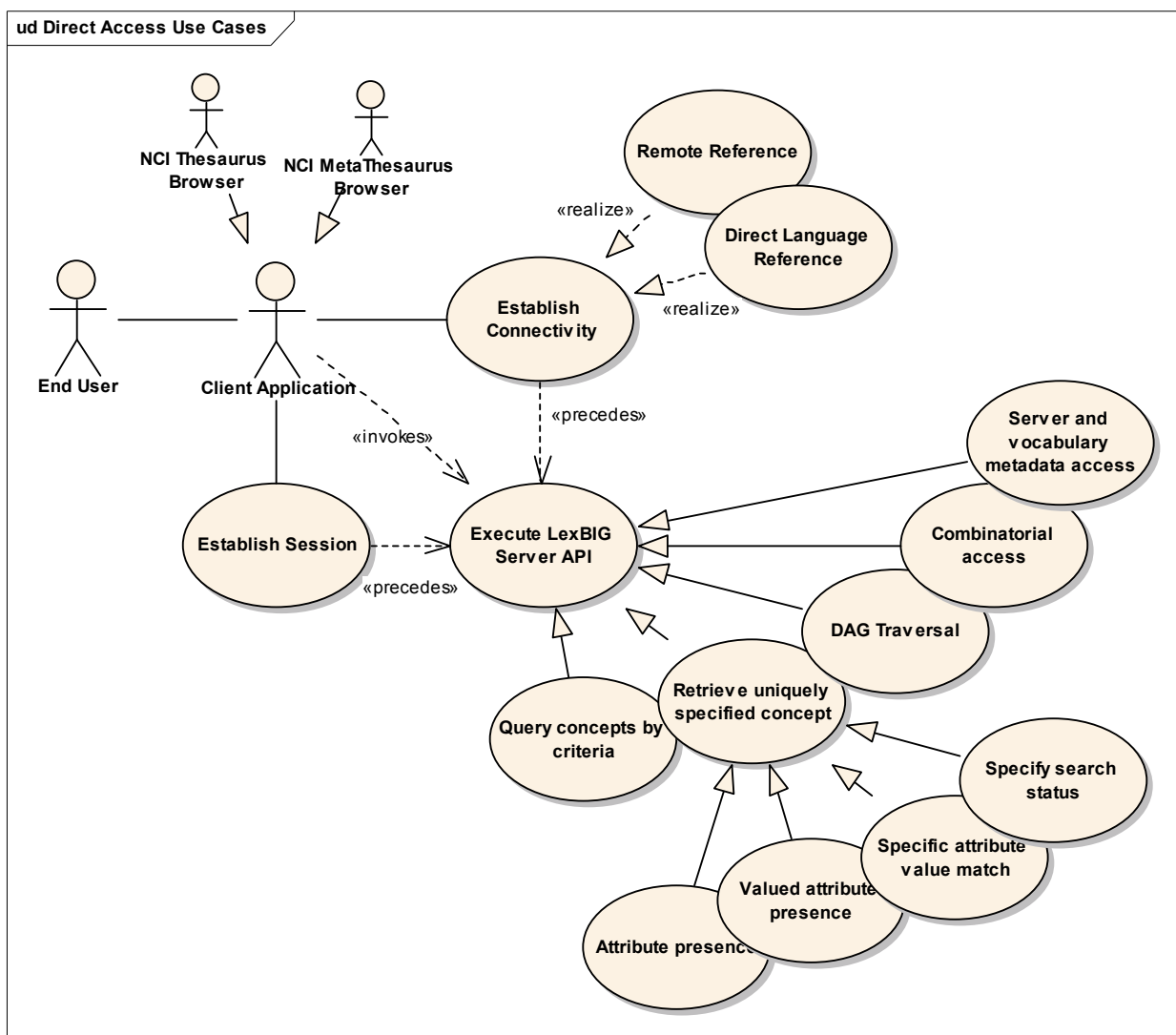
## 3.4.1.6 Submit spreadsheet to NCI

<b>Use Case ID</b>	<b>NCI UC 06</b>
<b>Primary Actor</b>	caBIG Modeler
<b>Secondary Actors</b>	
<b>Brief Description</b>	Actor submits model changes and mappings to NCI fore review
<b>Pre-conditions</b>	All class and attributes have been reviewed and checked for accuracy
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor submits to NCI for review</li> <li>2) Actor interacts with NCI modeler for appropriate definition</li> <li>3) NCI suggests mapping for missing or incorrect assignments</li> <li>4) Additional vocabulary may be created based on new definitions</li> </ol>
<b>Post Conditions</b>	Final spreadsheet file of class/attribute names with concept mapping
<b>Notes</b>	

### 3.4.2 Programmatic Access to LexBIG API

Programmatic access represents the primary pathway for accessing vocabulary content. This pathway can be used by a variety of external applications requiring vocabulary resources and functionality.

In addition to programmatic access through the caCORE EVS API, the LexBIG runtime environment supports a native API that may provide additional functionality not present in caCORE EVS. Invocation of the LexBIG API can be made using direct language binding or a service based method.



#### 3.4.2.1 Establish Connectivity

Use Case Id	PGM UC 01
Primary Actor	Client Application
Secondary Actors	End User
Brief Description	Actor establishes a connection to a LexBIG server (remote or direct language reference)
Pre-conditions	1) All necessary LexBIG server components are installed, loaded, and configured. 2) Actor incorporates necessary LexBIG connection code (e.g. client stubs for remote access, LexBIG Java implementation for direct language reference).
Flow of Events	1) Actor invokes appropriate method to instantiate a connection, as documented for the LexBIG client code that they are using. 2) Refer to use case <a href="#">Remote Reference</a> (PGM_UC_02) or <a href="#">Direct Language Reference</a> (PGM_UC_03) for details on how connection is realized.
Post Conditions	Actor has a connection to a caCORE server.

## 3.4.2.2 Remote Reference

<b>Use Case Id</b>	<b>PGM_UC_02</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	End User
<b>Brief Description</b>	LexBIG connection to a remote server.
<b>Pre-conditions</b>	PGM_UC_01 Actor has requested a connection to a remote server.
<b>Flow of Events</b>	1) Client library attempts to connect to the remote server. 2) Client library reports successful connection, or returns error to the user.
<b>Post Conditions</b>	Actor has a connection to a caCORE server.
<b>Notes</b>	

## 3.4.2.3 Direct Language Reference

<b>Use Case Id</b>	<b>PGM_UC_03</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	End User
<b>Brief Description</b>	Direct language reference to LexBIG implementation is established. Actor has requested a connection to a remote server.
<b>Pre-conditions</b>	PGM_UC_01
<b>Flow of Events</b>	1) LexBIG implementation is initialized. 2) Implementation reports initialization success or failure to User.
<b>Post Conditions</b>	Actor has a connection to a caCORE server.
<b>Notes</b>	

## 3.4.2.4 Establish Session

<b>Use Case Id</b>	<b>PGM_UC_04</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	End User
<b>Brief Description</b>	Actor requests a session from the LexBIG server, and sets session parameter values.
<b>Pre-conditions</b>	Actor has established connectivity (PGM_UC_01) Session Manager is available.
<b>Flow of Events</b>	1) Actor calls client API method to request a session. 2) Session Manager creates a session, and returns session information to client. ** 3) Client stores session information. 4) Actor calls method to change a session default value. 5) Client sends stored session information and new value into the session manager. a. Session Manager validates new value. b. Session Manager stores new value if valid, or returns an error if invalid.
<b>Post Conditions</b>	Actor has established a session.
<b>Notes</b>	If the Actor is using a direct language reference, the Client is the same thing as the LexBIG API implementation. If the Actor is using a remote reference, all actions are passed from the client to the LexBIG API implementation.  ** Extent and nature of session support will be determined on completion of LexBIG architecture.

## 3.4.2.5 Method Invocation

<b>Use Case Id</b>	<b>PGM_UC_05</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	End User
<b>Brief Description</b>	Actor calls a method from the LexBIG API.
<b>Pre-conditions</b>	Actor has established connectivity (PGM_UC_01) Optional – Actor has established session (PGM_UC_04)
<b>Flow of Events</b>	1) Actor provides parameters for and calls a LexBIG API client method. 2) The client passes the parameters to the LexBIG API Implementation. 3) Optional – if session information is present, it is also passed in.

	4) The implementation sends the parameters on to the proper method (see use cases PGM_UC_06 through PGM_UC_10) 5) The method returns the result to the Actor.
<b>Post Conditions</b>	Actor get results from calling a method.
<b>Notes</b>	If the Actor is using a direct language reference, the Client is the same thing as the LexBIG API implementation. If the Actor is using a remote reference, all actions are passed from the client to the LexBIG API implementation.

#### 3.4.2.6 Retrieve uniquely specified concept

<b>Use Case Id</b>	<b>PGM_UC_06</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	End User
<b>Brief Description</b>	Actor calls a method from the LexBIG API to retrieve a uniquely specified concept.
<b>Pre-conditions</b>	Actor has invoked the appropriate method (PGM_UC_05)
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Method receives necessary parameters to get a uniquely specified concept.</li> <li>2) Method validates parameters according to the LexBIG API specification (tbd) <ol style="list-style-type: none"> <li>a. If parameters are invalid, an error is returned.</li> </ol> </li> <li>3) LexBIG implementation is executed to get the desired result.</li> <li>4) Result is returned to the Actor.</li> </ol>
<b>Post Conditions</b>	Actor gets details (as specified by the LexBIG API) of the uniquely specified concept.
<b>Notes</b>	

#### 3.4.2.7 Query concepts by criteria

<b>Use Case Id</b>	<b>PGM_UC_07</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	End User
<b>Brief Description</b>	Actor calls a method from the LexBIG API to query concepts by specified criteria.
<b>Pre-conditions</b>	Actor has invoked the appropriate method (PGM_UC_05)
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Method receives necessary parameters to query concepts by criteria..</li> <li>2) Method validates parameters according to the LexBIG API specification (tbd) <ol style="list-style-type: none"> <li>a. If parameters are invalid, an error is returned.</li> </ol> </li> <li>3) LexBIG implementation is executed to get the desired result.</li> <li>4) Result is returned to the Actor.</li> </ol>
<b>Post Conditions</b>	Actor gets details (as specified by the LexBIG API) of the query.
<b>Notes</b>	

#### 3.4.2.8 DAG Traversal

<b>Use Case Id</b>	<b>PGM_UC_08</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	End User
<b>Brief Description</b>	Actor calls a method from the LexBIG API to traverse the Directed Acyclic Graph.
<b>Pre-conditions</b>	Actor has invoked the appropriate method (PGM_UC_05)
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Method receives necessary parameters to traverse the DAG in the requested direction, to the requested depth.</li> <li>2) Method validates parameters according to the LexBIG API specification (tbd) <ol style="list-style-type: none"> <li>a. If parameters are invalid, an error is returned.</li> </ol> </li> <li>3) LexBIG implementation is executed to get the desired result.</li> <li>4) Result is returned to the Actor.</li> </ol>
<b>Post Conditions</b>	Actor gets the requested nodes (as specified by the LexBIG API) from the DAG.
<b>Notes</b>	

#### 3.4.2.9 Combinatorial Access

<b>Use Case Id</b>	<b>PGM_UC_09</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	End User
<b>Brief Description</b>	Actor calls a method from the LexBIG API to execute a combination of other methods.

<b>Pre-conditions</b>	Actor has invoked the appropriate method (PGM_UC_05)
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Method receives necessary parameters to execute a combination of other methods.</li> <li>2) Method validates parameters according to the LexBIG API specification (tbd) <ol style="list-style-type: none"> <li>a. If parameters are invalid, an error is returned.</li> </ol> </li> <li>3) Multiple LexBIG methods are executed, using the results from one execution as input to the next method to get the desired result.</li> <li>4) Result is returned to the Actor.</li> </ol>
<b>Post Conditions</b>	Actor gets results (as specified by the LexBIG API) from the combination query.
<b>Notes</b>	<i>Combinatorial access is not considered in scope of LexBIG phase 1 deliverables.</i>

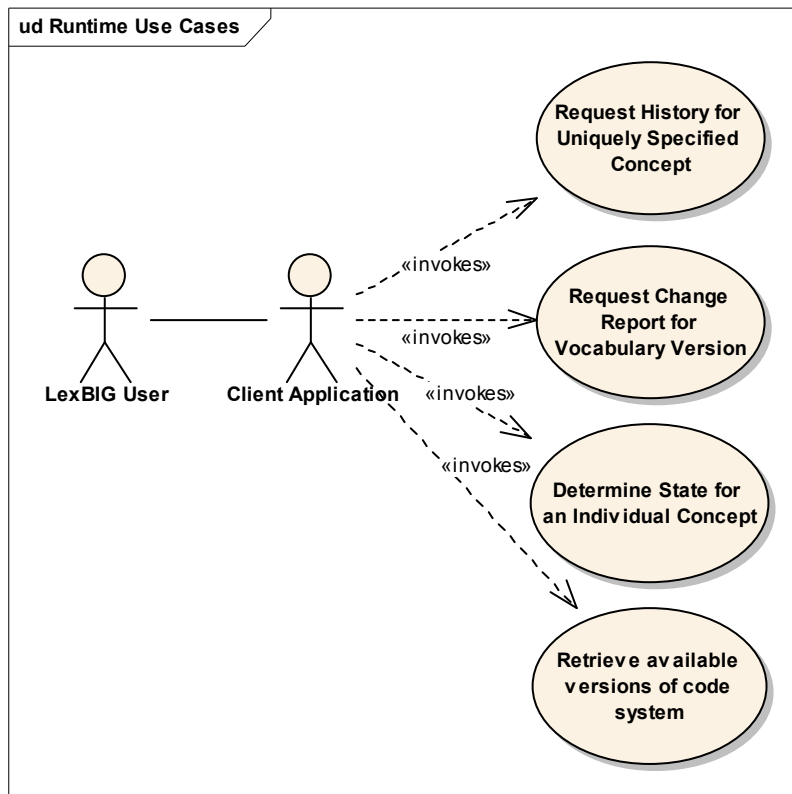
#### 3.4.2.10 Server and Vocabulary Metadata Access

<b>Use Case Id</b>	<b>PGM_UC_10</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	End User
<b>Brief Description</b>	Actor calls a method from the LexBIG API to get metadata.
<b>Pre-conditions</b>	Actor has invoked the appropriate method (PGM_UC_05)
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Method receives necessary parameters to get requested metadata.</li> <li>2) Method validates parameters according to the LexBIG API specification (tbd) <ol style="list-style-type: none"> <li>a. If parameters are invalid, an error is returned.</li> </ol> </li> <li>3) LexBIG implementation is executed to get the desired metadata.</li> <li>4) Result is returned to the Actor.</li> </ol>
<b>Post Conditions</b>	Actor gets results (as specified by the LexBIG API) from their metadata query.
<b>Notes</b>	

### 3.4.3 Access to History and Version Information

A number of scenarios justify the ability of a vocabulary server to provide access to vocabulary history and vocabulary versions. This functionality can be addressed with varying degrees of sophistication. Vocabulary history needs to be provided and maintained by the vocabulary providers. There is no computable mechanism to determine a concept history if not provided by vocabulary provider. Vocabulary versions have many uses for information retrieval supporting epidemiologic use cases. Versioning provides the ability to determine how a concept was used in a given time period. Ultimately, versioning would be best maintained by provider, but with separate namespace individual versions of a vocabulary can be published and accessed.

**NOTE:** Conversion and manipulation of history data is not considered in scope of LexBIG phase 1 deliverables.



#### 3.4.3.1 Request History for Uniquely Specified Concept

<b>Use Case Id</b>	<b>HST_UC_01</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	Client Application
<b>Brief Description</b>	Queries historical entries for a given concept.
<b>Pre-conditions</b>	Historical information has been imported from native format to the LexBIG repository.
<b>Flow of Events</b>	1) Actor invokes LexBIG API through client application specifying concept ID 2) System queries available history based the given parameters
<b>Post Conditions</b>	Actor gets results (as specified by the LexBIG API) from the query.
<b>Notes</b>	

#### 3.4.3.2 Request Change Report for Vocabulary Version

<b>Use Case Id</b>	<b>HST_UC_02</b>
<b>Primary Actor</b>	LexBIG User

<b>Secondary Actors</b>	Client Application,
<b>Brief Description</b>	Queries historical entries recorded for a specific version of a vocabulary.
<b>Pre-conditions</b>	Historical information has been imported from native format to the LexBIG repository.
<b>Flow of Events</b>	1) Actor invokes LexBIG API through client application specifying code system / version 2) System queries available history based on the given parameters
<b>Post Conditions</b>	Actor gets results (as specified by the LexBIG API) from the query.
<b>Notes</b>	

### 3.4.3.3 Determine State of Individual Concept

<b>Use Case Id</b>	<b>HST_UC_03</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	Client Application
<b>Brief Description</b>	Queries current state of a given concept.
<b>Pre-conditions</b>	
<b>Flow of Events</b>	1) Actor invokes LexBIG API through Client Application specifying concept ID and property names that correspond to the desired status (e.g. concept status, isActive flag, etc) 2) System queries assigned properties for the concept
<b>Post Conditions</b>	Actor gets results (as specified by the LexBIG API) from the query.
<b>Notes</b>	

### 3.4.3.4 Retrieve Available Versions of a Code System

<b>Use Case Id</b>	<b>HST_UC_04</b>
<b>Primary Actor</b>	LexBIG User
<b>Secondary Actors</b>	Client Application
<b>Brief Description</b>	Actor invokes the LexBIG API to query versions of a vocabulary stored in the LexBIG repository.
<b>Pre-conditions</b>	
<b>Flow of Events</b>	1) Actor invokes LexBIG API through client application specifying code system name. 2) System queries accessible versions of the vocabulary.
<b>Post Conditions</b>	Actor gets results (as specified by the LexBIG API) from the query.
<b>Notes</b>	

### 3.5 Security, Integrity, and Access Constraints

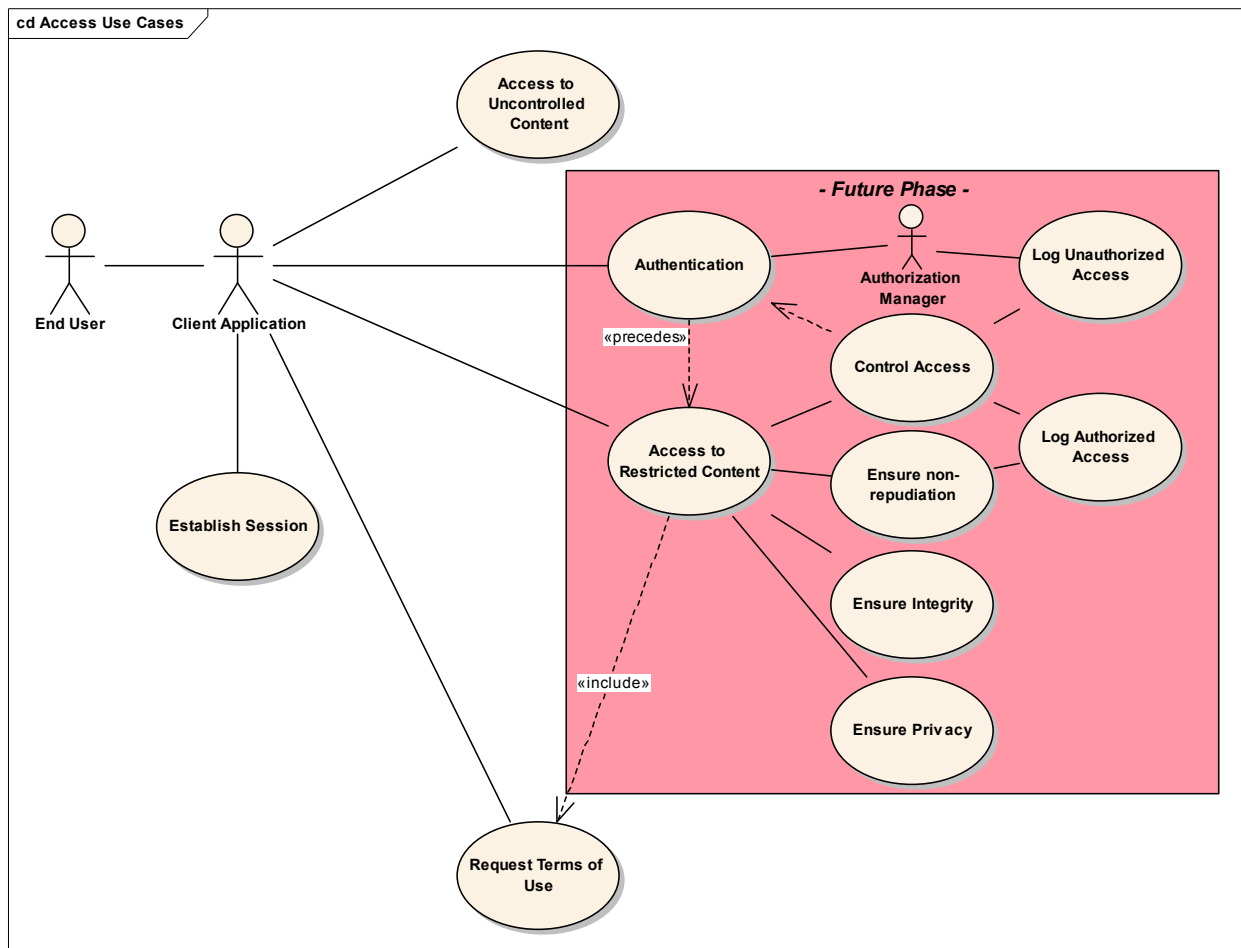
This section describes measures taken at runtime to ensure security and integrity for protected content. To support the goals of vocabulary federation the vocabulary server may need to protect some vocabulary content from unauthorized use. License and terms of usage are dependent on the vocabulary provider. Ultimately the license and terms of usage must be understood by the organization using the vocabulary. The vocabulary server will provide some mechanism for supporting security policy.

In consideration of baseline requirements, this document describes support of security policy in terms of active and passive models. In the passive model the server provides access to license terms as specified by the content provider. However, the server itself does not attempt to monitor or deny access based on this information. Instead, responsibility for appropriate access is to be delegated to the client application and end user. Client Applications that directly interact with the user are expected to monitor and present license/copyright agreements when requests are made against protected content. In this situation, a burden is also placed on the user to properly interpret and honor the terms once disclosed.

In contrast, the active model implies a more traditional token-based (e.g. password) authentication model. If an unauthorized user ID or token is presented on a request, authentication fails and any attempt to access protected content is denied outright by the server.

Phase 1 deliverables for the LexBIG project will incorporate support only for the passive security model. However, active authentication is also considered in the use case diagram so that the current design does not preclude its introduction in the future. Additional use cases will be added in the final draft to further describe both models and anticipated support within the LexBIG runtime environment.

The notion of a bound session is potentially utilized in either scenario.



### 3.5.1 Passive Model

In the passive model the server provides access to license terms as specified by the content provider. However, the server itself does not attempt to monitor or deny access based on this information.

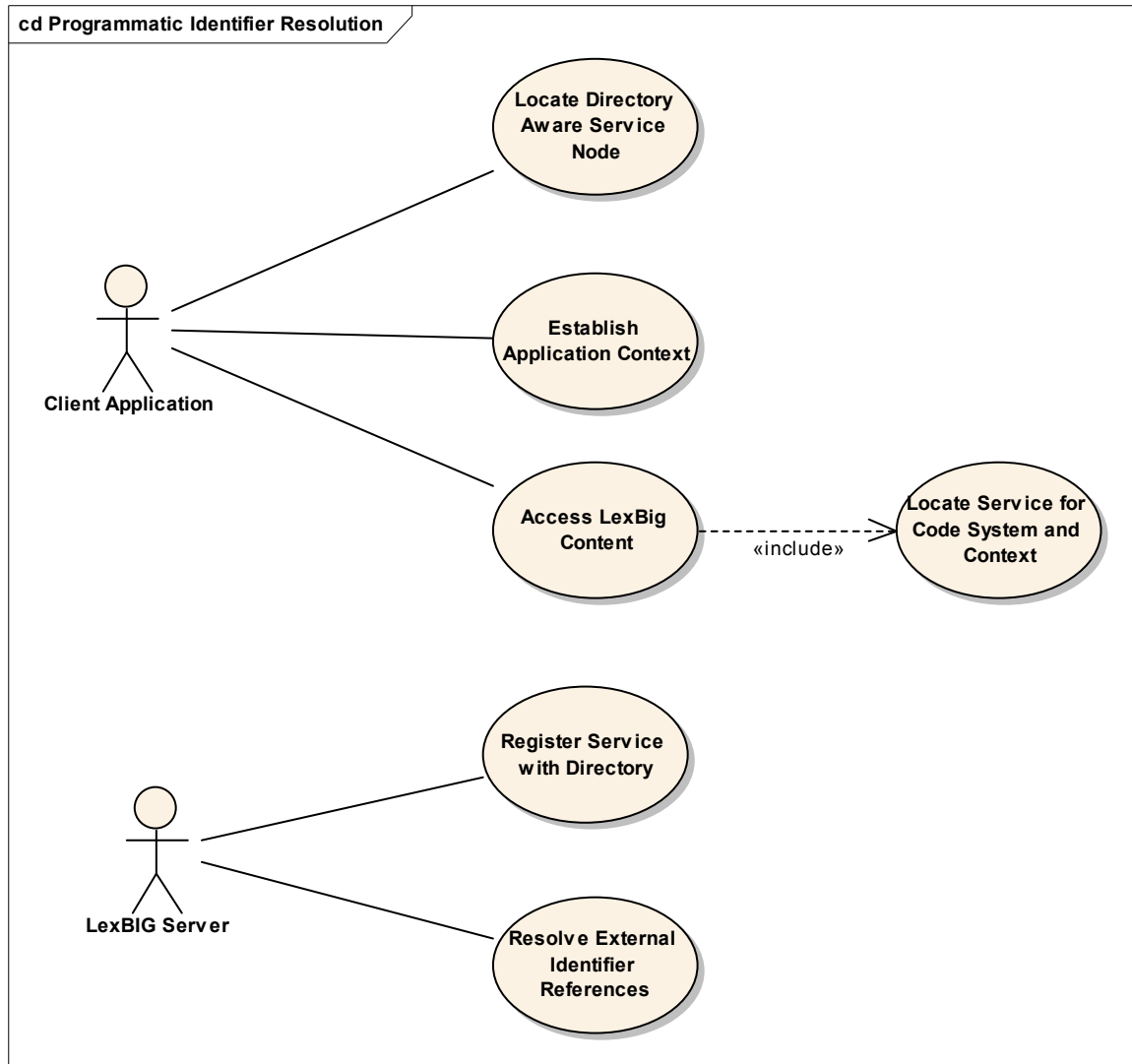
#### 3.5.1.1 Request Terms of Use

<b>Use Case ID</b>	<b>SEC_UC_01</b>
<b>Primary Actor</b>	End User
<b>Secondary Actors</b>	Client Application
<b>Brief Description</b>	With system adhering to a soft enforcement policy, access to vocabularies is not controlled by the system. Responsibility is delegated to the client program to determine terms of use and display a warning message to the user if appropriate.
<b>Pre-conditions</b>	Terms of use have been acquired from the content provider and registered for programmatic access.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor initiates request to access vocabulary through client program</li> <li>2) Client Application checks whether terms of use have been checked for the specified vocabulary within bounds of the current session</li> <li>3) If yes, no further action. Otherwise, terms of use are requested from the system.</li> <li>4) System retrieves license terms and returns them to client application.</li> <li>5) Client application evaluates response. If content is restricted, a dialog is displayed to the user calling out the terms and requesting confirmation to continue.</li> <li>6) User interprets license terms and confirms or cancels the request.</li> </ol>
<b>Post Conditions</b>	If the action is confirmed, requested information will be returned to the user. Otherwise, the client program returns nothing and awaits the next request.
<b>Notes</b>	

### 3.6 Identifier Resolution

This section covers the resolution of unique code system and concept identifiers within the LexBIG environment.

#### 3.6.1 Identifier Resolution



##### 3.6.1.1 Locate Directory Aware Service Node

Use Case ID	idRes UC_01
Primary Actor	Client Application
Secondary Actors	
Brief Description	LexBIG client establishes connection with LexBIG node that can provide access to directory services
Pre-conditions	
Flow of Events	The LexBIG client application uses an external mechanism to establish a connection with a LexBIG node that participates in the LexBIG directory services (LDS) for the grid
Post Conditions	Connection is established
Notes	

## 3.6.1.2 Establish Application Context

<b>Use Case ID</b>	<b>idRes UC 02</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	
<b>Brief Description</b>	LexBIG client records environment and context for subsequent directory resolution calls
<b>Pre-conditions</b>	Connection is established with LexBIG directory service
<b>Flow of Events</b>	<p>The client application supplies the necessary contextual information that will allow the directory service to unambiguously resolve code system and concept access requests. The context includes:</p> <ul style="list-style-type: none"> <li>• Authentication tokens</li> <li>• Required service protocol and formatting information</li> <li>• Location of the client within the network</li> <li>• Required performance characteristics</li> </ul>
<b>Post Conditions</b>	Context is established and is available for all subsequent directory and server calls
<b>Notes</b>	

## 3.6.1.3 Access LexBIG content

<b>Use Case ID</b>	<b>idRes UC 03</b>
<b>Primary Actor</b>	Client Application
<b>Secondary Actors</b>	
<b>Brief Description</b>	Map content access requests to the appropriate service node and issue forward responses to the calling application
<b>Pre-conditions</b>	Context has been established with directory services
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1. The client application requests resolution of a code system or concept code</li> <li>2. The LexBIG directory service uses the active directory and associated context to determine the appropriate service node</li> <li>3. The LexBIG directory service responds to the client with a redirection response, which contains the information needed to connect with the service in the protocol and format required by the client service</li> <li>4. The client establishes the connection with the appropriate service and begins to make API calls</li> </ol>
<b>Post Conditions</b>	Connection is established with the appropriate directory server
<b>Notes</b>	(Use the LDAP redirection services as a template for this model)

## 3.6.1.4 Register Service with Directory

<b>Use Case ID</b>	<b>idRes UC 04</b>
<b>Primary Actor</b>	LexBIG Server
<b>Secondary Actors</b>	
<b>Brief Description</b>	The server records the code systems, versions, protocols, formats and performance information with the LexBIG directory
<b>Pre-conditions</b>	Service knows how to connect with a LexBIG directory node
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1. The server establishes a connection with the LexBIG directory</li> <li>2. The server transmits the code systems, versions, protocols, formats and other information needed by the LexBIG directory</li> <li>3. The LexBIG directory periodically pings the server with requests for performance and statistics measurements. This ping also serves as a “stay-alive” ping to determine whether the server is still available for processing.</li> </ol>
<b>Post Conditions</b>	Directory is aware of the LexBIG node and has a connection with it
<b>Notes</b>	(The JNDI service spec has many useful details on how to do this)

## 3.6.1.5 Resolve External Identifier References

<b>Use Case ID</b>	<b>idRes UC 05</b>
<b>Primary Actor</b>	LexBIG Server
<b>Secondary Actors</b>	
<b>Brief Description</b>	The LexBIG server forwards requests for code systems and concept codes that it doesn't

---

	support locally to the LexBIG directory resolver
<b>Pre-conditions</b>	Directory is aware of the LexBIG node and has a connection with it
<b>Flow of Events</b>	<ol style="list-style-type: none"><li>1. The server receives a request to resolve a node that it doesn't handle locally</li><li>2. The server passes the client context and the node to be resolved to the LexBIG directory (see <b>idRES_UC_03</b>).</li><li>3. Depending upon the circumstances, the server either:<ol style="list-style-type: none"><li>a. acts as a "pass through" client and access the requested node information</li><li>b. returns a forward request to the client application if equipped to handle it</li></ol></li></ol>
<b>Post Conditions</b>	
<b>Notes</b>	Details about when to pass through and when to respond with a forward request need to be resolved.

## 3.7 **Collaboration**

This section covers use cases derived based on input from key organizations or initiatives within the caBIG community working in association with the LexBIG project.

### 3.7.1 **caTIES Retrieval Application**

A research investigator (user) is formulating search criteria to support a research question. The search criteria can represent a single search condition or a complex Boolean expression of unions, intersections, or exceptions. To build a query condition the caTIES application supports two modes of creation.

The user can browse the vocabulary to identify concepts of interest. Based on the type of vocabulary content explicit relationships may be navigable to aid the user in finding the query.

Alternately, the user can enter a search expression. This search expression can be checked for spelling and used to search the vocabulary to identify an appropriate matching concept. Searching the vocabulary based on the search expression will look for exact and lexically similar concepts based on the description of the concept. Once appropriate vocabulary concepts have been identified the user can assemble the concepts into a query expression using standard Boolean logic.

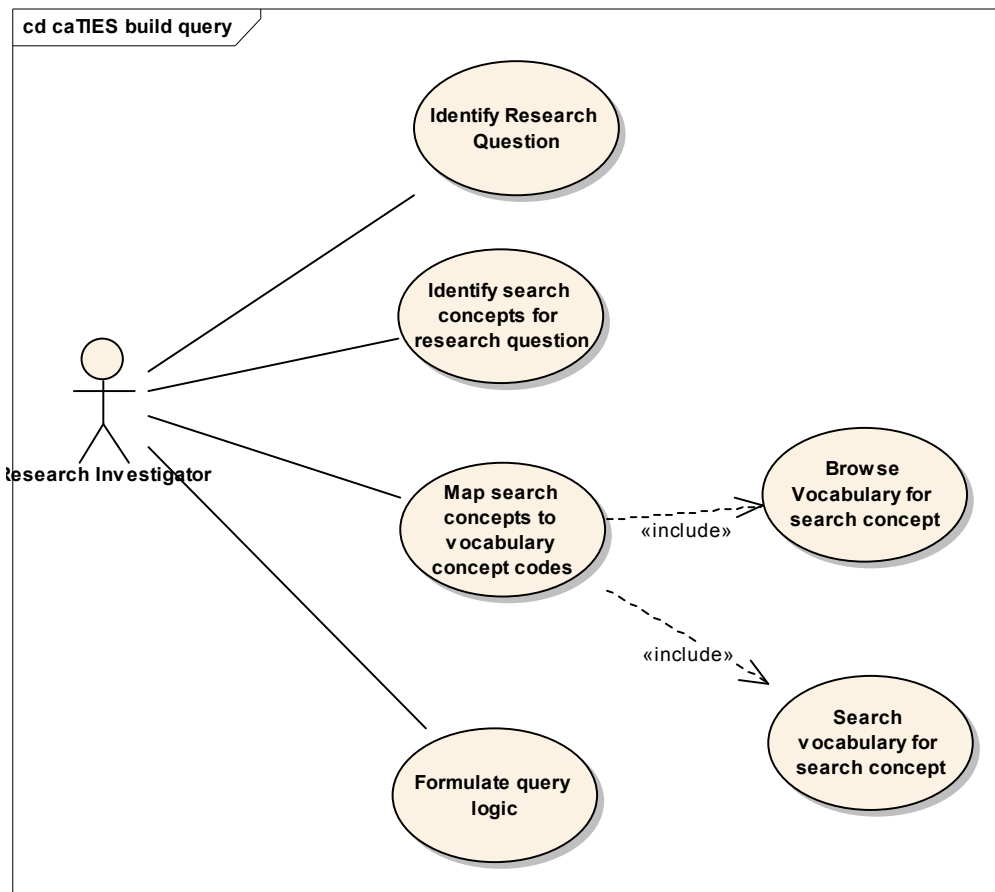
#### **Specific Notes:**

The caTIES application is a Java Web Start Java that provides a user interface for an investigator to construct a query with specific search criteria to identify pathology reports of interest. The application supports distributed queries, where the construct query submitted to one or more distributed sites that maintain the indexed pathology reports. Results are merged and presented as from one data source. This query distribution uses web services technology based on OGSA-DAI. Eventually, caGRID will be used for remote data access. Approximately, one million pathology reports are annotated for caTIES.

Based on some usability feedback, two modes of query construction are available to the user. The first mode is labeled dashboard, which provides basic term searching with automated best mapping to concept(s). This provides simplified “web-search-like” interface. The second mode is labeled diagram. This mode allows the user to construction a query visually using a graph-like syntax. JGraph is the specific technology used for display. The user can also browse the vocabulary to identify specific concepts for searching. The diagram mode allows the user to construct a complex query of union, intersections, and negation. One temporal constraint is allowed per query.

The search method can be in the form of keyword or phrase or concept code. The user can use a search concept function to enter a search term. This term is used by the vocabulary server to map to an appropriate concept/node in the vocabulary hierarchy. Once a match is found the associated siblings are also returned. The term to concept mapping uses the same program logic as the caTIES annotation module, relying on a version of metemap transfer MMTx from National Library of Medicine. This version of the algorithm and vocabulary content is locally maintained. To find the appropriate siblings the application using the caBIO-EVS interface using the NCI vocabulary servers. This separation of vocabulary content and methods of access raises a challenge for vocabulary version synchronization of annotation, query construction, and vocabulary browsing, currently requiring three or more instances of vocabulary content and services.

Currently, queries can be stored containing the concept used, but no mechanism or re-execution based on vocabulary version is provided. This functionality is desired and may require multiple versions of the vocabulary to be available as well as concept history. Pathology reports will be annotated with different versions of the vocabulary over time. Current annotations may yield different concept mappings from previous versions as additional concepts are added or modified.



### 3.7.1.1 Identify Research Question

<b>Use Case ID</b>	<b>caTIES_UC_01</b>
<b>Primary Actor</b>	Research Investigator
<b>Secondary Actors</b>	NA
<b>Brief Description</b>	Actor formulates a research question based on research hypothesis and research dilemma
<b>Pre-conditions</b>	
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Identify research dilemma</li> <li>2) Review, explore, and determine related or prior knowledge</li> <li>3) Define the research hypothesis</li> <li>4) Review, explore, and determine related or prior knowledge</li> <li>5) Formulate a set of research questions</li> <li>6) Formulate a set of retrieval parameters to fulfill research question</li> </ol>
<b>Post Conditions</b>	A set of research questions and retrievals to fulfill the answer or partial answer to the research question.
<b>Notes</b>	

### 3.7.1.2 Identify Search Concepts for Data Retrieval

<b>Use Case ID</b>	<b>caTIES_UC_02</b>
<b>Primary Actor</b>	Research Investigator
<b>Secondary Actors</b>	NA
<b>Brief Description</b>	Actor determines the medical concepts required to fulfill the data retrieval request
<b>Pre-conditions</b>	Knowledge of vocabulary scheme used for source data
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Identify subject areas for research question</li> <li>2) Identify search concepts in subject area</li> <li>3) Determine the concept granularity requirements</li> </ol>
<b>Post Conditions</b>	A list of search concepts
<b>Notes</b>	See Figure

## 3.7.1.3 Map Search Concepts to Vocabulary Concept Codes

<b>Use Case ID</b>	<b>caTIES_UC_03</b>
<b>Primary Actor</b>	Research Investigator
<b>Brief Description</b>	Actor determines the mapping of search concepts to appropriate vocabulary concepts or data elements
<b>Pre-conditions</b>	Knowledge of vocabulary concepts used for source data
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) For each search concept identify vocabulary concept(s)</li> <li>2) Identify concept specificity for search concept</li> <li>3) Find vocabulary concept(s) representing search concept</li> </ol>
<b>Post Conditions</b>	A list of vocabulary concepts
<b>Notes</b>	Includes Browse ( <a href="#">caTIES_UC_04</a> ) and Search ( <a href="#">caTIES_UC_05</a> ) use cases.

## 3.7.1.4 Browse Vocabulary for Search Concept

<b>Use Case ID</b>	<b>caTIES_UC_04</b>
<b>Primary Actor</b>	Research Investigator
<b>Brief Description</b>	Actor browses vocabulary contents to find the concept(s) representing search concept.
<b>Pre-conditions</b>	Actor has some domain knowledge of subject matter of vocabulary
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor reviews the content of the vocabulary</li> <li>2) Actor traverses navigable relationships to aid in finding vocabulary concept(s)</li> <li>3) Actor selects vocabulary concept(s) that match the search concept</li> </ol>
<b>Post Conditions</b>	A list of vocabulary concepts
<b>Notes</b>	

## 3.7.1.5 Search Vocabulary for Search Concept

<b>Use Case ID</b>	<b>caTIES_UC_05</b>
<b>Primary Actor</b>	Research Investigator
<b>Brief Description</b>	Actor enters a search expression (word or phrase) represent the lexical form of the vocabulary concept description.
<b>Pre-conditions</b>	Actor has some domain knowledge of subject matter of vocabulary
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor enters search expression in a the form of words representing lexical form of concept description</li> <li>2) Actor reviews vocabulary concept(s) that match the search expression</li> <li>3) Actor selects vocabulary concept(s) that match the search concept</li> </ol>
<b>Post Conditions</b>	A list of vocabulary concepts
<b>Notes</b>	See Figure 1 – caTIES 2.0 Search Concepts

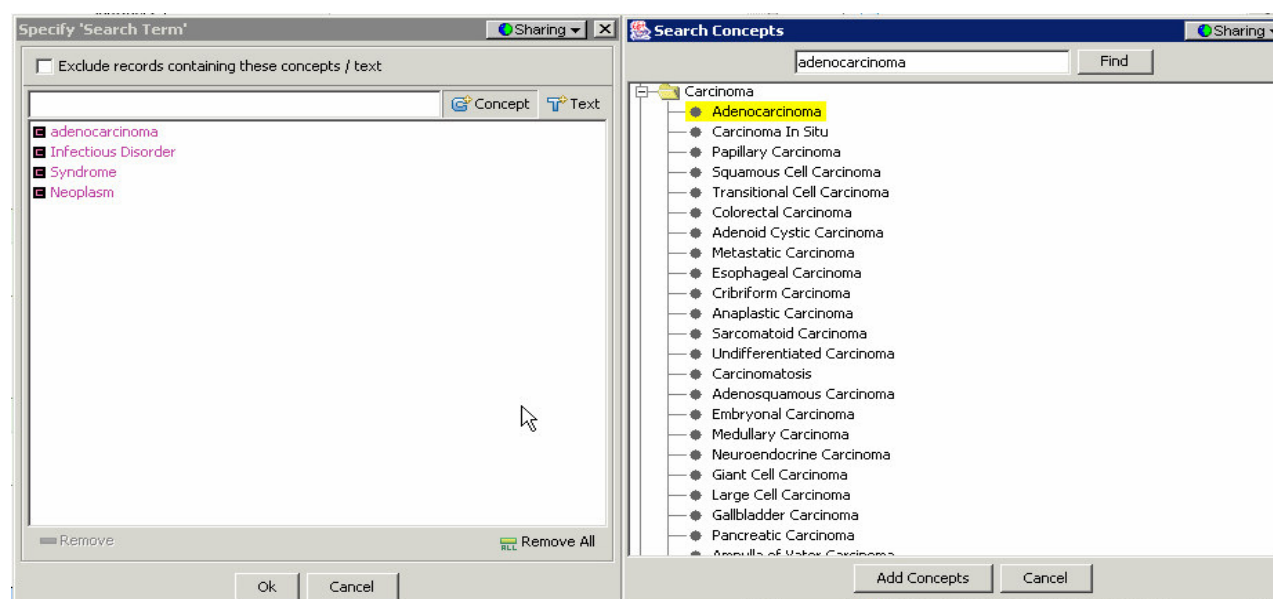


Figure 1 – caTIES 2.0 Search Concepts

## 3.7.1.6 Formulate Query Logic

<b>Use Case ID</b>	<b>caTIES_UC_06</b>
<b>Primary Actor</b>	Research Investigator
<b>Secondary Actors</b>	NA
<b>Brief Description</b>	Actor applies logic operators to the search conditions represented by vocabulary concepts
<b>Pre-conditions</b>	Actor has some familiarity of Boolean logic
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Actor selects vocabulary concepts to be grouped for a Boolean function (and, or, not)</li> <li>2) Actor repeats grouping and logic until search results match the retrieval question</li> <li>3) Actor selects vocabulary concept(s) that match search concept</li> </ol>
<b>Post Conditions</b>	A query
<b>Notes</b>	See Figure 2 – caTIES Query Formulation

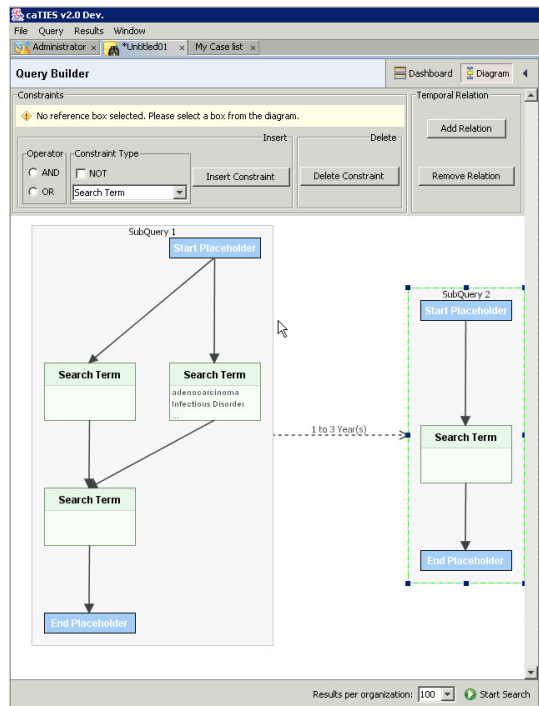
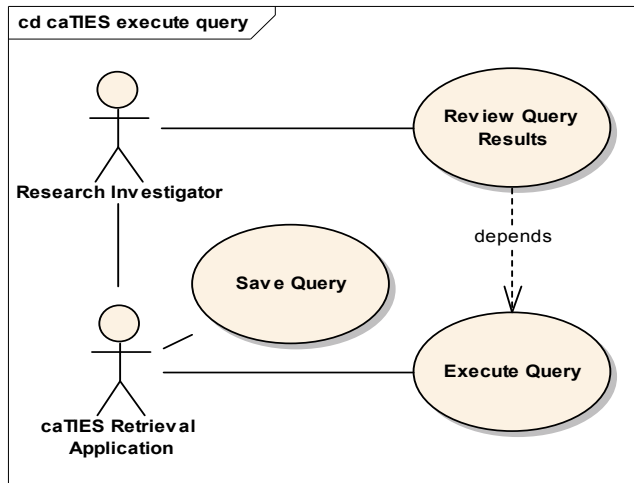


Figure 2 – caTIES Query Formulation

### 3.7.2 caTIES Execute Query

This use case is initiated by the research investigator once a retrieval query has been created. The research investigator submits the query to the caTIES system. Once the query is completed, the investigator reviews the results (documents) for appropriate level of detail and correctness as required of the research question. If necessary, the investigator may iteratively modify or refine the query to achieve applicable results. The investigator can save the query with search conditions for later reference and use.



#### 3.7.2.1 Execute Query

Use Case ID	caTIES_UC_07
Primary Actor	Research Investigator
Secondary Actors	caTIES Retrieval Application
Brief Description	Actor used retrieval application to submit query to caTIES system.
Pre-conditions	Query has been defined.
Flow of Events	1) Submit query to caTIES system 2) Review acknowledgement from system
Post Conditions	Acknowledgement of query submission

#### 3.7.2.2 Save Query

Use Case ID	caTIES_UC_08
Primary Actor	Research Investigator
Secondary Actors	caTIES Retrieval Application
Brief Description	Actor saves a query to the caTIES retrieval environment for later reference, use, or refinement.
Pre-conditions	Query has been defined.
Flow of Events	1) Save query 2) Review save acknowledgement
Post Conditions	Saved query

#### 3.7.2.3 Review Query results

Use Case ID	caTIES_UC_09
Primary Actor	Research Investigator
Secondary Actors	caTIES Retrieval Application
Brief Description	Actor reviews the query results (documents) to determine appropriateness of the query definition as it relates to the research question.
Pre-conditions	Query has been defined and submitted to the caTIES system
Flow of Events	1) Query results are displayed 2) Actor reviews results for relevance
Post Conditions	None

### 3.7.3 caTIES Annotation

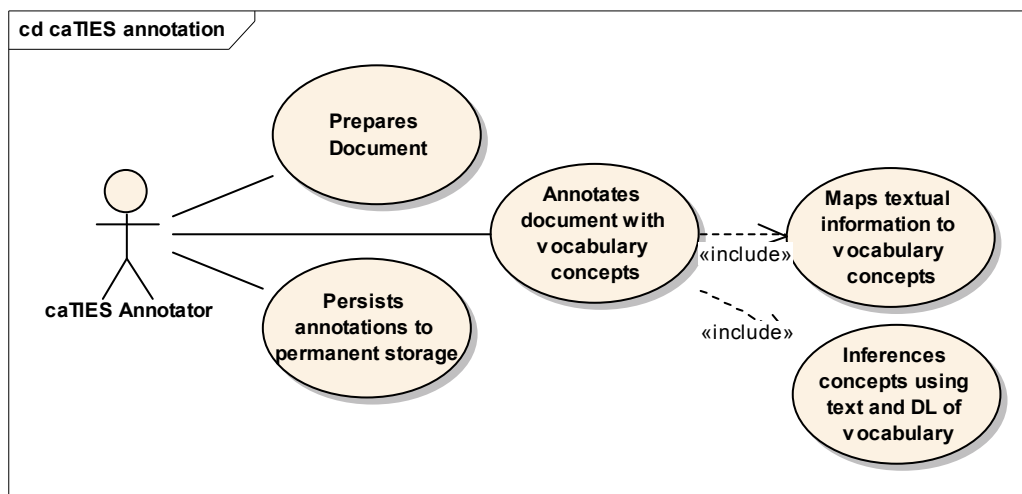
This use case is initiated by creation or modification of a clinical document. The document is prepared and formatted by the annotation software for subsequent analysis. The annotator assigns vocabulary concepts via lexical and semantic techniques based on the textual content. The vocabulary concepts and related attributes are stored in a database system.

#### Specific Details:

The caTIES annotation module performs textual processing on pathology reports. The caTIES module produces a set of vocabulary concepts with attributes that conceptually represent the content of the pathology report. The types of vocabulary concepts identified are dependent on the source vocabulary, but generally include disease, stage, grade, histology, and anatomy. The main method of annotation leverages the MetaMap Transfer (MMTx) software written by the National Library of Medicine. caTIES customizes the vocabulary used for MMTx operation. MMTx and vocabulary content is configured locally for performance and selectivity. New versions of the NCI Metathesaurus and NCI thesaurus trigger a rebuild process for the MMTx instance. This process is semi-automated using software processes with periodic human intervention. Overall a vocabulary update requires one to two day duration, not necessarily human effort. Support for a version delta that could be applied to an existing vocabulary server instance is desirable.

Additional processing of concept attribution, such as semantic type or negation is applied after the MMTx process. There was some interest in applying a rules engine, such as Jess, to vocabulary concepts, report text, report elements, and vocabulary description logic (DL) to inference additional concepts. DL access for vocabulary services was noted.

The annotation process produces a set of concepts for storage and later retrieval by the caTIES retrieval application. There was an issue regarding concept uniqueness where concept unique identifiers (CUI) were reused or changed meaning. The exact mechanism of concept persistence for caTIES is not fully understood, but should include concept identifier, coding scheme, and coding scheme version. This point identifies a potential need for allowing multiple versions of the vocabulary to be accessible. This highlights the importance of concept history.



#### 3.7.3.1 Prepares Document

Use Case ID	caTIES_UC_10
Primary Actor	caTIES Annotator
Secondary Actors	
Brief Description	Actor prepares clinical document for processing
Pre-conditions	New or changed clinical document
Flow of Events	1) Process and format clinical document
Post Conditions	Successful receipt and validation of clinical document structure
Notes	HL7 Clinical Document Architecture for semi-structured clinical documents

## 3.7.3.2 Annotate document with vocabulary concepts

<b>Use Case ID</b>	<b>caTIES_UC_11</b>
<b>Primary Actor</b>	caTIES Annotator
<b>Secondary Actors</b>	Vocabulary Server
<b>Brief Description</b>	Actor processes textual information by mapping text to vocabulary concepts.
<b>Pre-conditions</b>	Vocabulary content is accessible
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Text is partitioned into meaningful units, e.g. sections, paragraphs, sentences</li> <li>2) Text is mapped to vocabulary concepts</li> <li>3) Post-processing for additional, e.g. negation</li> </ol>
<b>Post Conditions</b>	List of vocabulary concepts and attributes
<b>Notes</b>	This is based on Metamap Transfer (MMTx) software written by National Library of Medicine (NLM)

## 3.7.3.3 Maps textual information to vocabulary concepts

<b>Use Case ID</b>	<b>caTIES_UC_12</b>
<b>Primary Actor</b>	caTIES Annotator
<b>Secondary Actors</b>	Vocabulary Server
<b>Brief Description</b>	Actor process clinical document and maps text to vocabulary concepts
<b>Pre-conditions</b>	Vocabulary content is accessible
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Text is partitioned into meaningful units, e.g. sections, paragraphs, sentences, tokens</li> <li>2) Lexical matching e.g. acronym or abbreviation expansion</li> <li>3) Part of Speech Markup</li> <li>4) Syntax parser</li> <li>5) Lexical variant generation</li> <li>6) Candidate concept identification</li> <li>7) Candidate evaluation and mapping</li> <li>8) Post-processing e.g. negation</li> </ol>
<b>Post Conditions</b>	List of vocabulary concepts and attributes
<b>Notes</b>	Based on NLM MMTx process flow <a href="http://mmtx.nlm.nih.gov/design/">http://mmtx.nlm.nih.gov/design/</a>

## 3.7.3.4 Inference concepts based using text and DL of vocabulary

<b>Use Case ID</b>	<b>caTIES_UC_13</b>
<b>Primary Actor</b>	caTIES Annotator
<b>Secondary Actors</b>	Vocabulary Server
<b>Brief Description</b>	Actor process clinical document and maps text to vocabulary concepts
<b>Pre-conditions</b>	Vocabulary content is accessible. Description logic of vocabulary is accessible.
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Vocabulary concepts are identified via textual mapping</li> <li>2) Using description logic of vocabulary, text, and mapped concepts use a rule engine (e.g. Jess – Java Expert System Shell) to inference additional vocabulary concepts.</li> <li>3) Post-processing</li> </ol>
<b>Post Conditions</b>	List of vocabulary concepts and attributes
<b>Notes</b>	

## 3.7.3.5 Persists annotation to permanent storage

<b>Use Case ID</b>	<b>caTIES_UC_14</b>
<b>Primary Actor</b>	caTIES Annotator
<b>Secondary Actors</b>	Data Storage
<b>Brief Description</b>	Actor saves vocabulary concepts and attributes to a database.
<b>Pre-conditions</b>	Query has been defined and submitted to caTIES system
<b>Flow of Events</b>	<ol style="list-style-type: none"> <li>1) Vocabulary concepts and attributes are stored to database</li> <li>2) Acknowledgement of transaction</li> </ol>
<b>Post Conditions</b>	Vocabulary concepts and attributes stored in database
<b>Notes</b>	

## 4 NOTES

---

Mayo continues to actively monitor the requirements of Jackson Labs and the Cancer Research Center of Hawaii as integrators of the Mouse Anatomy and Nutrition Ontology, respectively. We have not identified unique LexBIG requirements or use cases for these efforts at this time. However, additional use cases will be introduced as appropriate and as time permits based on the ongoing feedback provided by these organizations.

## Appendix A – Acronym List

Term/ Abbreviation	Description
API	Application Programmatic Interface
caTIES	Cancer Text Information Extraction System
CRCH	Cancer Research Center Hawaii
DAG	Directed Acyclic Graph
DBA	Database Administrator
EVS	Enterprise Vocabulary Services
LSI	LexBIG Service Index
NCICB	NCI Center for Bioinformatics
OBO	Open Biomedical Ontologies
OWL	Ontology Web Language
PM	Project Manager
RM	Requirements Manager
RRF	Rich Release Format
RTM	Requirements Traceability Matrix
SCM	Software Configuration Management
SDLC	Software Development Lifecycle
SDP	Software Development Plan
SE	Software Engineering
SEPG	Software Engineering Process Group
SM	Software Manager
SOP	Standard Operating Procedure
SPI	Software Process Improvement
SQA	Software Quality Assurance
SW	Software
TBD	To Be Determined
TM	Test Manager
UMLS	Unified Medical Language System
UPMC	University of Pittsburgh Medical Center